The Pragmatic Programmers

TURING 图灵程序设计丛书

Hello, Android

Introducing Google's Mobile Development Platform, Fourth Edition

Android 基础教程（第4版）

【美】Ed Burnette 著 袁国忠 译

通过游戏制作，入门Android开发，
实现在Google Play Store快速发布应用



中国工信出版集团 人民邮电出版社 POSTS & TELECOM PRESS

数字版权声明

图灵社区的电子书没有采用专有客户端，您可以在任意设备上，用自己喜欢的浏览器和PDF阅读器进行阅读。

但您购买的电子书仅供您个人使用，未经授权，不得进行传播。

我们愿意相信读者具有这样的良知和觉悟，与我们共同保护知识产权。

如果购买者有侵权行为，我们可能对该用户实施包括但不限于关闭该帐号等维权措施，并可能追究法律责任。

Ed Burnette

资深软件技术专家，拥有30多年的软件开发经验。他是SAS高级计算机实验室的联合创始人和高级研究员，还是Planet Android（www.planetandroid.com）网站的创办人和ZDNet的专栏作家。除本书外，他还出版了*Google Web Toolkit: Taking the Pain out of Ajax*和*Eclipse IDE Pocket Guide*等著作。

袁国忠

自由译者；2000年起专事翻译，主译图书，偶译新闻稿、软文；出版译著40余部，其中包括《C++ Prime Plus中文版》《CCNA学习指南》《CCNP ROUTE学习指南》《面向模式的软件架构：模式系统》《Android应用UI设计模式》《风投的选择：谁是下一个十亿美元级公司》等，总计700余万字；专事翻译前，从事过三年化工产品分析和开发，做过两年杂志和图书编辑。

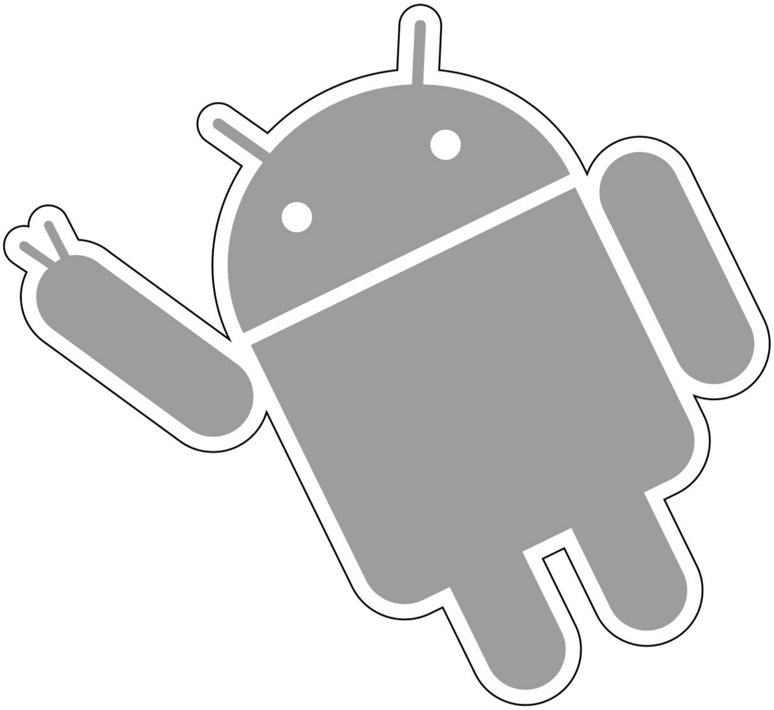
图灵程序设计丛书

Hello, Android

Introducing Google's Mobile Development Platform, Fourth Edition

Android基础教程（第4版）

【美】Ed Burnette 著　袁国忠 译



人民邮电出版社
北　京

图书在版编目（CIP）数据

Android基础教程 : 第4版 / (美) 伯内特
(Burnette,E.) 著 ; 袁国忠译. -- 北京 : 人民邮电出
版社, 2016.1
（图灵程序设计丛书）
ISBN 978-7-115-40860-0

Ⅰ. ①A… Ⅱ. ①伯… ②袁… Ⅲ. ①移动电话机—应
用程序—程序设计—教材 Ⅳ. ①TN929.53

中国版本图书馆CIP数据核字(2015)第258184号

内 容 提 要

本书是一部关于 Android 开发的基础教程，以由浅入深、循序渐进的方式讲解了 Android 程序设计的核心概念和技术。本书不仅结合井字游戏开发案例形象生动地讲解了 Android 生命周期、用户界面、简单的数据存储等基础知识，而且还深入探讨了外部通信、基于位置的服务、内置 SQLite 数据库等高级主题。每章最后都提供了“快速阅读指南”，通过它可以迅速找到所需信息，并高效地完成工作。

本书适合所有移动开发人员阅读。

◆ 著　　[美] Ed Burnette
译　　袁国忠
责任编辑　朱　巍
责任印制　杨林杰

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京　　　　　印刷

◆ 开本：800×1000　1/16
印张：11.75
字数：291千字　　　　2016年1月第1版
印数：1－4 000册　　　2016年1月北京第1次印刷

著作权合同登记号　图字：01-2015-5416号

定价：49.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316

反盗版热线：(010)81055315

广告经营许可证：京崇工商广字第 0021 号

版权声明

致　　谢

本书得以付梓，需要感谢的人很多。感谢前几版读者提出极佳的建议；感谢编辑Susannah Pfalzer对细节的关注；感谢Craig Castelaz、Javier Collado、Eric Hung、Edward Ingram、Chris Johnson、Howard Koslow、Helen Li、Irakli Nadareishvili、Jan Nonnen、Jason Pike、Mike Riley、Sam Rose、Loren SandsRamshaw、Carlos Santana、Charley Stran和Stephen Wolff提出宝贵的审阅意见；尤其要感谢Lisa、Chris和Michael一直以来的耐心和支持。

前　　言

Android是一款用于手机和平板电脑的开源操作系统，由Google及其合作伙伴和其他参与者开发。使用Android的手机和其他移动设备现已超过10亿部，这让它占据了头号应用开发平台的宝座。无论你是业余爱好者还是专业程序员，无论你从事Android开发只是出于好玩还是为了赚钱，都需要更深入地学习。本书旨在帮助你起步。

Android的独特之处

市面上的其他移动平台还有很多，包括iOS、Windows、Tizen、Firefox OS等。为何大家会选择Android？它有什么独特之处呢？

虽然Android的一些功能并不新颖，但它是第一个兼具如下特点的环境。

- 基于Linux的免费开源平台：手机制造商对其钟爱有加，因为他们可以对这个平台进行定制，而无需支付版权费。开发人员也喜欢它，因为他们知道这个平台不受制于任何可能破产或被收购的厂商。
- 基于组件的架构（其灵感来自于Internet混搭技术）：能够以不同于开发人员最初设想的方式使用应用的组成部分，甚至可以将内置组件替换为改进版本。这在移动领域发起了新一轮的创意运动。
- 大量现成的服务：基于位置的服务使用GPS或基站三角学定位，让你能够根据设备位置定制用户体验。五脏俱全的SQL数据库能够让你好好地利用本地存储，设备只需时不时地联网同步即可。浏览器和地图视图可以直接嵌入到应用中。这些功能都能使应用的功能得到进一步的提升，同时还降低了开发成本。
- 自动管理应用的生命周期：多重安全保障能够将程序彼此隔离，从而提高了系统的稳定性。最终用户不必关心哪些应用处于活动状态，也无需关闭一些程序以便运行其他程序。Android针对电量和内存有限的设备进行了彻底优化，这是以前的平台没有尝试过的。
- 高品质图形和声音：流畅而平滑的2D和3D加速图形支持新的游戏和商业应用。内置的编码/解码器支持常见的行业标准音频和视频格式，包括H.264（AVC）、MP3和AAC。
- 到各种既有和未来硬件的移植性：所有程序都使用Java编写，并由Android的ART预先编译器或Dalvik虚拟机执行，因此代码可移植到ARM、x86和其他体系结构。支持各种输入

方法，如键盘、游戏手柄、触摸、电视遥控、鼠标和跟踪球。可针对任何屏幕分辨率和朝向定制用户界面。

Android在移动应用与用户交互方面进行了新的尝试，并为这种交互提供了技术支持。但在整个Android生态系统中，最重要的还是开发人员为之编写的软件。本书将帮助你在这方面开好头。

针对的读者

本书是为想快速熟悉Android平台的新手编写的。只需几分钟，你就能安装好开发工具并编写第一个程序。阅读完本书后，你将能够编写引人入胜的完整应用。更重要的是，你将能够找到并看懂Android开发旅程中所需要的高级材料。

要阅读本书，你必须明白基本的Java编程概念，如类、方法、作用域和继承，还必须知道`import`、`static`、`final`、`public`和`this`等Java关键字的含义。如果你不明白我在说什么，建议先阅读一本Java入门书，如：

- *Java Precisely* [Ses05]
- 《Head First Java（中文版）》（*Head First Java*）[SB05]
- 《Effective Java中文版》（*Effective Java*）[Blo08]
- 《Java程序设计语言》（*The Java Programming Language*）[AGH05]
- 《Java技术手册（第6版）》（*Java in a Nutshell*）[EF14]

你不需要具备任何移动设备软件开发经验。实际上，如果你有这样的经验，最好将其抛诸脑后，因为Android是如此地与众不同，刚接触它时最好是没有任何的成见。然而，如果你具备IntelliJ IDEA、Eclipse或Visual Studio等集成开发环境（IDE）的使用经验，这些经验将派上用场。

涵盖的内容

本书分为四大部分，大致是按从简单主题到复杂主题，或者说从Android常见方面到不那么常见的方面来安排的。

我们利用几章的篇幅开发了一个示例：Android终极版井字棋游戏。通过逐渐给这个游戏添加功能，你将学习Android编程的很多方面，包括用户界面、多媒体以及Android活动和片段的生命周期。

第一部分首先将简要地介绍Android，你将学习如何安装Android模拟器以及如何使用IDE来编写第一个程序。接下来将介绍几个重要的概念，如Android应用的生命周期。Android编程与你习惯的方式稍有不同，请务必掌握这些概念后再接着往下阅读。

第二部分讨论Android用户界面：显示、输入、多媒体和动画。你编写的很多程序都将用到这些功能。

第三部分更深入地探讨Android平台，包括如何让应用与多种Android设备和版本兼容，以及如何将应用发布到Google Play Store。

第四部分讨论一些较高级的主题，包括嵌入HTML页面、访问Web服务、使用Google Play服务以及使用内置的SQLite数据库存储数据。

本书的最后是附录和参考文献。其中，附录介绍了Android和Java Standard Edition（SE）的差别。

新增内容

本版经过了修订，以支持Android 4.1到Android 5.1的所有版本。本书所说的Android“现代”版，指的是Android 4.1（Jelly Bean）和更高版。

旧版本

Android 2.3（Gingerbread）是最后一个只支持手机的版本。Android 3.0（Honeycomb）是重要的分水岭，但只支持平板电脑，因此采用它的设备十分有限（但在Google内部所处的地位无疑是最高的）。Android 4.0（Ice Cream Sandwich）同时支持手机和平板电脑，但几乎没有在Android 3.0的基础上新增任何功能。

Android 4.1 的新特性

在Android 4.1中，Google为改善可用性和性能做了很大努力。这个计划的代号为Project Butter。Google添加了测量系统速度和效率的新方法，进而优化了处理时间的使用效率[①]。

Android 4.2 的新特性

Android 4.1取得了巨大成功。受此鼓励，Google决定在接下来的两个版本中保留原来的代号。Android 4.2（Jelly Bean MR1）在改善性能的道路上继续前行，同时还新增了多用户支持、使用Miracast标准以无线方式将屏幕投射到远程显示器的功能[②]。

① http://d.android.com/sdk/api_diff/16/changes.html
② http://d.android.com/sdk/api_diff/17/changes.html

Android 4.3 的新特性

Android 4.3（Jelly Bean MR2）的重点是安全性。它将SE（Security Enhanced）Linux作为底层操作系统，并新增了权限设置（restricted profile）功能，让设备所有者能够给不同用户设置不同的权限。它还是第一个支持OpenGL ES 3.0的版本[①]。

Android 4.4 的新特性

在Android 4.4（KitKat）中，最重要的新特性是，将基于WebKit的WebView替换成了Chrome浏览器[②]使用的Chromium引擎。

Android 4.4W的新特性

Android Wear（KitKat for Watches）是用于智能手表的操作系统，为支持可穿戴设备[③]，必须做一些修改和修复。

Android 5.0 的新特性

Android 5.0（Lollipop）最显著的变化是引入了一种新的设计语言——Material Design。在内部，之前所有版本都使用的Dalvik VM被替换为了ART。ART依靠提前编译来改善性能。最后，开启了Project Volta计划[④]，其目标与Project Butter相同，但旨在延长电池的续航时间，而不是改善性能。

Android 5.1 的新特性

Android 5.1（Lollipop MR1）支持多个SIM卡，新增了通过Google Play分发运营商应用的功能。另外，摒弃了`AndroidHttpClient`类以及org.apache.http包中大量的类[⑤]。

如果我没记错，L后面是M、N、O和P。只要采纳本书的建议，你所编写的程序只需做少量修改（甚至无需修改），就能支持未来的Android版本。第8章介绍了如何创建支持多个版本的程序。

要了解各种Android版本的最新市场份额，请参阅Android Device Dashboard[⑥]。本书的所有示例都针对Android 4.1~5.1版进行了测试。

---

① http://d.android.com/sdk/api_diff/18/changes.html
② http://d.android.com/sdk/api_diff/19/changes.html
③ http://d.android.com/sdk/api_diff/20/changes.html
④ http://d.android.com/sdk/api_diff/21/changes.html
⑤ http://d.android.com/sdk/api_diff/22/changes.html
⑥ http://d.android.com/resources/dashboard/platform-versions.html

本书不支持Android 4.1之前的版本，因为它们占据的市场份额很小并且还在不断萎缩。本书也没有过多考虑Android 5.1支持的自定义，因为编写本书时，使用Android 5.1的设备还不多。为确保短小精悍，本书只探讨大多数Android程序都涉及的主题。

在线资源

本书的配套网站[①]提供了如下资源。

- 本书所有示例程序的完整源代码以及声音和图像等资源。
- 列出本版错误的勘误表（但愿它始终是空的）。
- 让你能够直接与作者和其他Android开发人员交流的论坛（但愿它内容多多）。

只要你觉得合适，可以在你的应用中随便使用这些源代码。请注意，如果你阅读的是电子书，也可以单击代码清单前面的三角形来直接下载源代码文件。

快速阅读指南

大多数作者都希望读者逐字阅读其著作，但我知道你不会这样做，你只想学习完成手头任务所需的知识，等以后需要完成其他任务时，再回过头来阅读相关的内容。有鉴于此，我尽力提供了相关帮助，以免你迷路。

本书每章都以“快速阅读指南”结束，为下一章的内容提供了指南，让你在不按顺序阅读时知道接下来该阅读的内容；同时还列出了图书和在线文档等其他资源，让你能够更深入地了解相关的主题。

现在就开始阅读吧。第1章将引导你动手创建一个非常简单的Android程序；第2章则会回过头来介绍Android基本概念和理念；第3章将介绍一个井字棋示例，并深入探讨用户界面——大多数Android程序最重要的部分。

你的终极目标是，将应用提交到Play Store进行销售或供人免费下载，因此第9章将演示如何完成这个最后的步骤。

① http://pragprog.com/book/eband4

目　　录

第三部分 创造性思维

第四部分 进阶

第一部分

Android简介

第1章 快速入门

Android是一种激动人心的开源移动平台，它像手机一样无处不在，得到了Google以及其他一些开放手机联盟成员（如三星、HTC、中国移动、Verizon和AT&T等）的支持，因而不能不加以学习，否则你承担不起为此付出的代价。

好在Android开发入门很容易，即使没有Android手机都没关系，只需有一台可供安装Android SDK和设备模拟器的计算机即可。

本章首先介绍如何安装所有的开发工具，然后再创建一个可运行的应用——Android版"Hello, World"。如果你并非Android新手，那么可以快速浏览本章，也可跳过本章，直接进入第2章。

1.1 安装工具

Android软件开发包（SDK）适用于Windows、Linux和Mac OS X，使用它开发的应用可部署到任何Android设备。

要进行Android开发，必须先安装Java、IDE和Android SDK。

1.1.1 Java开发包 7.0+

首先，需要安装Java开发包（JDK）。所有Android开发工具都需要它，在编写程序时将使用Java语言。要求安装JDK 7或8。

> **注意** Mac用户可跳过这一小节，因为Android Studio会自动安装合适的JDK版本（如果你没有安装的话）。然而，有人提出，在Mac上存在JDK版本不匹配的问题。如果你遇到了错误，可参阅Stack Overflow网站[①]的故障排除技巧[②]。

① http://stackoverflow.com/questions/24472020
② http://stackoverflow.com/questions/16636146

仅安装Java运行环境（JRE）还不够，还必须安装完整的Java开发包。建议从Oracle下载网站[①]下载最新的Java SE 8 JDK进行更新。

还需设置环境变量`JAVA_HOME`，使其指向JDK安装位置。具体如何设置取决于所使用的操作系统。例如，在Windows 7中，可以单击“开始”按钮，右击“计算机”并选择“属性”，再单击“高级系统设置”，然后单击“环境变量”按钮，再单击“系列变量”列表下方的“新建”按钮，然后在“变量名”文本框中输入JAVA_HOME，并在“变量值”文本框中输入JDK安装目录。最后，单击“确定”按钮关闭所有的窗口并保存设置。

要核实JDK版本是否正确无误，可打开一个shell窗口（在Windows中，要打开shell窗口，可单击“开始”按钮，输入cmd并按回车键），并执行如下命令。后面是我执行这些命令时得到的输出：

```
C:\> java -version
java version "1.8.0_31"
Java(TM) SE Runtime Environment (build 1.8.0_31-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.31-b07, mixed mode)

C:\> echo %JAVA_HOME%
C:\Program Files\Java\jdk1.8.0_31
```

你应看到类似的输出，其中的版本号为1.7或更高。

1.1.2　Android Studio

接下来，需要安装Java开发环境（如果还没有安装的话）。建议使用Android Studio，因为它是免费的，并得到了打造Android的Google开发人员的采用和支持。

务必使用最新的beta版或生产版。请访问Android Studio下载页面[②]，并单击Download Android Studio按钮。

注意　如果不想使用Android Studio（大千世界什么人都有），也可使用NetBeans和Eclipse等IDE，它们都有相应社区的支持。如果你非常老派，根本不想使用IDE，也可只使用命令行工具[③]。本书假设你使用的是Android Studio，如果不是这样，就需要做必要的调整。

---

① http://www.oracle.com/technetwork/java/javase/downloads
② http://d.android.com/sdk
③ http://d.android.com/tools/help

Eclipse怎么了？

直到最近，大部分Android开发人员使用的依然是Eclipse IDE[1]和Android Development Tools。2013年5月，Google推出了Android Studio——一款新的开发环境，它基于JetBrains开发的IntelliJ IDEA[2]。

Android Studio的最大不同在于，它使用的是Gradle编译系统。Android Studio还提供了很多新功能，如经过重大改进的WYSIWYG编辑器、支持使用相同代码生成多种配置。Eclipse依然得到了支持，但大多数新开发都将在Android Studio中进行。

1. http://www.eclipse.org
2. http://www.jetbrains.com/idea

下载并安装Android Studio后，启动它，并按屏幕指示进行操作。对于所有设置，都接受默认的标准值，即不断地单击Next按钮，并最终单击Finish按钮。下载并安装所需的一切可能需要几分钟，最终你将看到图1-1所示的界面。



图 1-1

这意味着已经成功地安装了Android Studio，可以开始开发了。

别忘了，Android Studio在不断地发展变化，因此你看到的界面可能会与本书所展示的稍有

不同。在新版本中，默认文件名和目录也可能不同。如果遇到这样的差别，请相应地调整操作，并报告到本书的在线论坛[①]。

真是讨厌！好在你只需这样做一次。现在万事俱备，是时候编写第一个程序了。

1.2 创建第一个程序

Android Studio自带了多个示例程序，即模板。下面使用其中一个模板来创建一个简单的“Hello, Android”程序。这只需几秒钟就能完成。请准备秒表。准备好了吗？出发！

选择Start a new Android Studio project，打开New Project对话框。

将依次出现4个界面。其中，第一个界面要求给出应用的名称和存储位置，如图1-2所示。

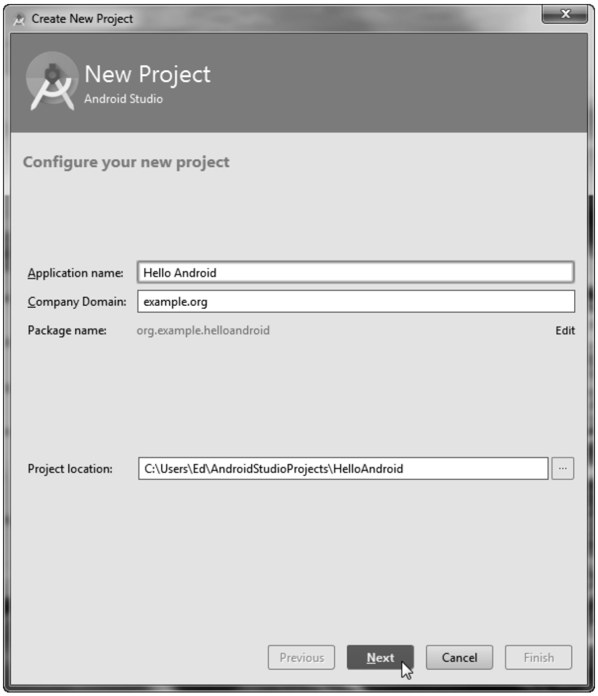


图 1-2

将应用名设置为Hello Android，并将公司域名设置为example.org，Android Studio会自动填写其他内容。单击Next按钮继续执行。

第二个界面提示指定适用的Android版本，如图1-3所示。

① http://pragprog.com/book/eband4



图　1-3

选择复选框Phone and Tablet，并将Minimum SDK指定为API 16: Android 4.1 (Jelly Bean)。这一步很重要，请务必确保选择了正确的版本。接下来，单击Next按钮。

第三个界面要求选择要添加的示例活动的类型，如图1-4所示。

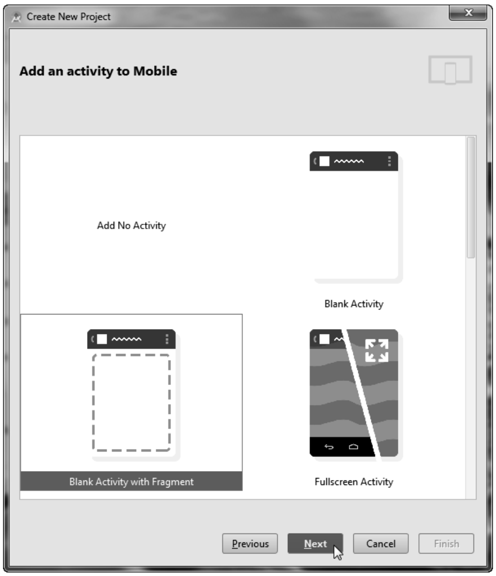


图　1-4

选择Blank Activity with Fragment，并单击Next按钮。

最后一个界面要求指定活动名和其他信息，如图1-5所示。

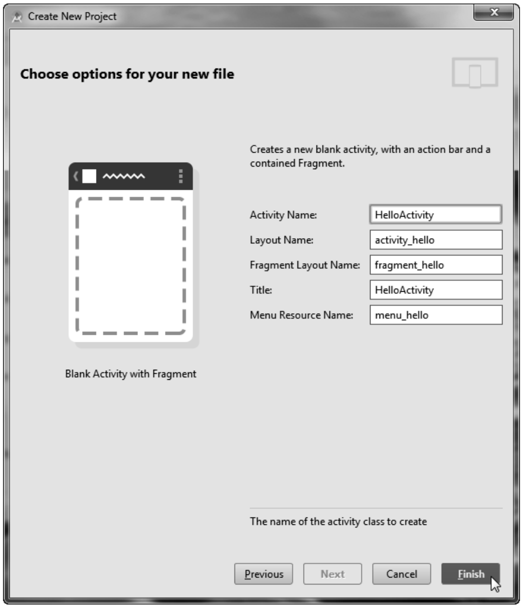


图 1-5

将活动名（Activity Name）改为HelloActivity，其他内容将自动被填写完成。

在本书后面的示例中，为了节省时间，我们将采用如下简化方式指出这些新建项目设置。

- 应用名：Hello Android
- 公司域名：example.org
- 尺寸：Phone and Tablet
- 最低SDK：API 16，Android 4.1 (Jelly Bean)
- 添加活动：Blank Activity with Fragment
- 活动名：HelloActivity

填写完最后一个界面中的相关内容后，单击Finish按钮。IDE将创建指定的项目，其中包含一些默认文件。接下来，IDE将对其进行编译和打包，为执行做好准备。

注意 在显示fragment_hello.xml的编辑器中，如果出现了有关渲染问题（Rendering Problems）的错误消息，不用管它，只需将窗口关闭即可。这是Android Studio中一个已知的bug。

向导将程序编写好了，你需要做的只是尝试运行它。下面，首先在Android模拟器中运行它。

1.3　在 Android 模拟器中运行

模拟器是一个程序，它运行在一种硬件上，却可以模拟另一种硬件。使用Android模拟器，几乎可在台式机上创建任何平板电脑、手机和可穿戴设备的虚拟版本。

要运行Android程序，可选择菜单Run➤Run 'app'，也可单击工具栏上的Run按钮，如图1-6所示。

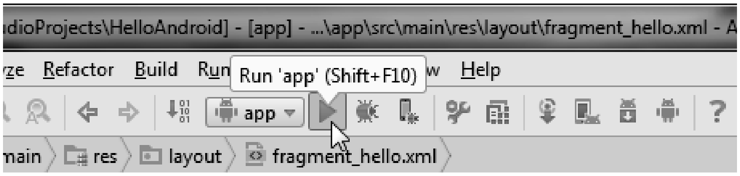


图　1-6

过一会儿后，将出现Choose Device对话框，如图1-7所示。

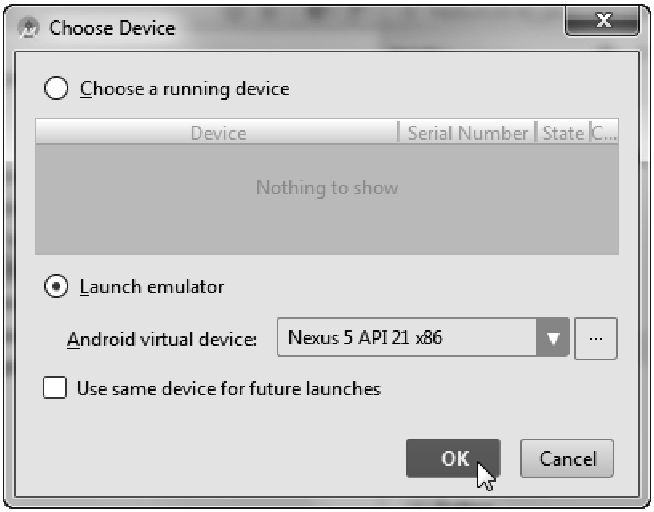


图　1-7

确定选择了Launch emulator，并指定了Android虚拟设备（AVD）的名称。单击OK按钮，运行程序。

将打开Android模拟器窗口并启动Android操作系统。首次这样做时，这可能需要一两分钟，请耐心等待。如果屏幕被锁定，请按说明轻扫鼠标以解锁。

Android Studio将程序的副本发送给模拟器并执行它。此时将出现应用界面，这说明“Hello, Android”程序正在运行，如图1-8所示。



图　1-8

如果几分钟后模拟器还没有出现，或者看起来像停止了一样，可能是由于你的计算机不与Intel硬件加速兼容。为解决这种问题，可新建一个AVD，并指定ARM处理器而不是Intel x86。更详细的信息请参阅8.1节。另一种解决方案是使用Genymotion模拟器[①]。

就这么简单！祝贺你编写了第一个Android程序。

1.4　在实际设备上运行

在开发期间，要在物理设备（如Nexus 5）上运行Android程序。做法几乎与在模拟器中运行时相同。在使用Android 4.2或更高版本的设备中，需要先启用开发者模式，即启动应用“设置”，再选择“关于手机”或“关于平板电脑”，然后轻按“版本号”7次（这是Android开发者提供的一个复活节彩蛋）；之后再启用USB调试，即依次选择“开发者选项”➤“调试”➤“USB调试”。

在计算机上安装Android USB设备驱动程序（如果没有安装的话，仅Windows系统需要这样做），然后使用设备自带的USB电缆将设备连接到计算机。

① http://www.genymotion.com

首次安装USB驱动程序时可能比较棘手。Using Hardware Devices页面[1]提供了最新的设备驱动程序及其安装说明。如果出现一个消息框，询问是否允许USB调试（其中还显示了你的计算机的RSA密钥指纹），请选择复选框Always allow from this computer，再单击OK按钮。

以后再运行应用时，该设备将出现在Choose Device窗口中。可以同时运行多个模拟器和设备，并在每次运行应用时都选择要使用的设备或模拟器；也可以选择复选框Use same device for future launches。如果设备没有出现在列表中，通常意味着，要么USB驱动程序有问题，要么针对的Android版本不对。

应用准备就绪后，要将其发布给其他人使用。这需要执行一些额外的步骤，将在第9章进行详细介绍。

缩短周转时间

启动模拟器需要很长时间。可以这样想象一下：开启手机时，它需要像其他任何计算机系统一样启动，而关闭模拟器就像是关闭手机并取出电池一样。因此，请不要关闭模拟器！

在Android Studio运行期间，应始终打开模拟器窗口。这样，下次启动Android程序时，Android Studio将注意到模拟器正在运行，因此只需要将程序发送给它去运行即可。

1.5　其他步骤

为节省时间，前面省略了两个步骤，下面来对其加以介绍。

1.5.1　检查更新

Android Studio还不是很成熟，修改频率比Android SDK高得多。你下载的版本可能不同于本书使用的版本，可能还会有一些独特之处。

有鉴于此，建议令其自动检测更新，以便立即下载并安装新的更新。还可以随时手动检查更新，为此可选择菜单Help➤Check for Update。

1.5.2　添加SDK包

Android Studio安装程序包含Android SDK和基本开发工具。然而，随着学习的不断深入，你可能会发现还需要其他工具。要获取这些工具，可运行Android SDK Manager。

① http://d.android.com/tools/device.html

在Android Studio中，选择菜单Tools➤Android➤SDK Manager。管理器将显示一个可用组件列表，其中包括文档、平台、插件库和USB驱动程序，如图1-9所示。

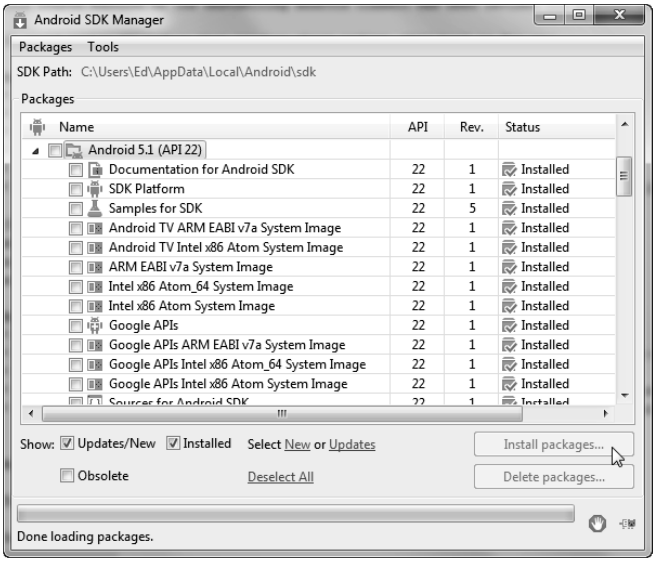


图　1-9

建议安装如下组件的最新版本。

- Android SDK Tools：软件开发包。
- Android SDK Platform-tools：低级工具，如adb（Android Debug Bridge）。
- Android SDK Build-tools：编译工具。
- Android 5.1（API 22）（或更高版本）：安装最高版本的所有组件。

安装类别Extras下的如下组件。

- Android Support Repository：gradle需要它。
- Android Support Library：用于与较旧的Android版本兼容。
- Google Play services：增值库，包含很多不错的功能。
- Google Repository：gradle需要它。
- Google USB Driver（仅Windows）：让你能够在实际设备上运行和调试程序。
- Intel x86 Emulator Accelerator：提高模拟器速度的插件包。

选择要安装的所有组件后，单击Install按钮。安装可能需要很长时间才能完成。在系统询问是否接受许可协议时，务必接受所有的许可协议（有时有多个许可协议）。出现有关重启SDK Manager的消息时，让系统重启就是了。如果出现HTTPS SSL错误，请在Android SDK Manager中选择菜单Tools➤Options，再选择复选框Force https:// sources to be fetched using http://。

1.6　快速阅读指南

拜Android Studio所赐，创建一个Android程序骨架只需几秒钟。第3章将开始充实这个骨架，让它变成一个货真价实的应用——一个井字游戏。本书的很多章都将使用这个示例来演示Android API。

在着手编写这个游戏之前，请先花几分钟时间阅读第2章。掌握活动和生命周期等基本概念后，本书余下的内容理解起来将会容易得多。

开发Android程序时，并非一定要使用Android Studio，不过还是强烈建议使用它。如果你以前从未使用过IntelliJ，可能应该花点时间阅读JetBrains网站[①]的IntelliJ IDEA快速入门指南。需要特别注意的是快捷键，因为只要牢记几个快捷键，就可以节省大量时间。

① http://www.jetbrains.com/idea/documentation

第2章 重要概念

对Android是什么有大致了解后，来看看其工作原理。对于Android的有些组成部分，如Linux内核和SQL数据库，你可能很熟悉，而对于其他组成部分，如Android的应用生命周期概念，你可能是完全陌生的。

要编写行为良好的Android应用，必须深入了解这些重要概念。因此，如果你只想阅读本书的某一章，那就阅读本章吧。

2.1 总览

先来看看总体系统架构——Android开源软件栈的重要分层和组件。图2-1说明了Android的总体情况，请仔细研究。



图2-1 Android系统架构

每层都使用它下面各层提供的服务。下面从最底层开始，来着重介绍Android的各层。

2.1.1　Linux 内核

Android以经过实践检验的可靠Linux内核为基础。Linux是Linus Torvalds于1991年开发的，目前被广泛用于从腕表到超级计算机的各种设备中。Linux为Android提供了硬件抽象层，让Android能够移植到各种平台之上。

在内部，Android使用Linux来提供内存管理、进程管理、联网和其他操作系统服务。Android用户根本看不到Linux，而你编写的程序通常也不会直接使用Linux调用。然而，作为开发人员，你必须知道Linux内核的存在。

在开发期间所使用的某些工具会与Linux进行交互。例如，命令`adb shell`①用于打开一个Linux shell，让你能够在其中输入要在设备上运行的其他命令。你可以通过它审视Linux文件系统、查看活动进程等，但要受到安全限制的约束。

2.1.2　原生库

内核的上一层包含Android原生库。这些共享库都是使用C或C++编写的，针对Android设备使用的硬件架构进行编译，并由厂商预装。

下面是一些最重要的原生库。

- Surface Manager：绘图命令并不直接在屏幕上绘画，而是被保存到列表中。这些命令列表与来自其他窗口的命令列表合并，一起生成用户看到的综合效果。这让系统能够创建各种有趣的效果，如透明窗口和令人着迷的过渡。
- 2D和3D图形：在Android中，可在用户界面中同时使用二维和三维元素。所有元素都由硬件转换为3D绘图列表并进行渲染，以最大限度地提高显示速度。
- 多媒体编码解码器：Android可播放各种格式的视频和音频，包括AAC、AVC（H.264）、H.263、MP3和MPEG-4。
- SQL数据库：Android包括一个轻量级的SQLite数据库引擎②——Firefox和Apple iPhone使用的数据库。可以在应用中使用它来实现持久化存储。
- 浏览器引擎：为快速显示HTML内容，Android使用了Chromium库③。这是Google Chrome浏览器使用的引擎，与Apple Safari浏览器和Apple iPhone使用的引擎很像。

这些库本身并非应用，仅供更高层的程序调用。要编写和部署原生库，可使用原生开发工具包（NDK，Native Development Toolkit）。原生开发不在本书的讨论范围内，如果你对此感兴趣，可阅读在线资料④。

---

① http://d.android.com/tools/help/adb.html
② http://www.sqlite.org
③ http://www.chromium.org
④ http://d.android.com/tools/sdk/ndk

2.1.3 Android 运行时

在内核之上，还有Android运行时，包括运行环境和核心Java库。运行环境使用Dalvik或ART，具体使用哪个取决于Android版本。

Dalvik是Google的Dan Bornstein设计并编写的一款虚拟机（VM）。我们编写的代码会被编译为独立于机器的指令——字节码，然后由移动设备上的Dalvik VM执行。

ART（Android Runtime）是一款超前的编译器，Android 5.0（Lollipop）用它取代了Dalvik。ART在将应用安装到Android设备时会将其编译成机器码。相比于Dalvik，ART可以提高程序的运行速度，但代价是安装时间更长。

Dalvik和ART都是Google推出的准兼容性Java实现，都针对移动设备进行了优化。在Android开发中，所有代码都使用Java编写，并由Dalvik或ART运行。

请注意，Android自带的核心Java库不同于Java Standard Edition（Java SE）库和Java Mobile Edition（Java ME）库，但它们有很多相同的地方。附录对Android Java库和标准Java库做了比较。

2.1.4 应用框架

在原生库和运行时之上，是应用框架层。它提供了用于创建应用的高级构件。这个框架是随Android预安装的，你可以根据需要对其进行扩展，在其中添加自己的组件。

> **拥抱并扩展**
>
> Android的一个独特而强大之处是，所有应用都处于一个公平竞争的环境中。也就是说，系统应用与第三方应用一样，都使用相同的公有API。如果你愿意，甚至可以让Android用你的应用替换标准应用。

下面是应用框架最重要的组成部分。

- 活动管理器：它控制着应用的生命周期（参见2.3节），并维护一个用于用户导航功能的通用后退栈。
- 内容提供器：这些对象封装了要在应用之间共享的数据，如通讯录。详情请参阅2.2.6节。
- 资源管理器：资源指的是程序中除代码之外的其他所有东西，详情请参阅2.2.7节。
- 位置管理器：Android设备始终知道其身处何方，详情请参阅第12章。
- 通知管理器：以不唐突的方式将短信、约会、接近提示、外族入侵等事件告知用户。

2.1.5　应用和服务

在前面的Android架构图中，最顶层是应用和服务层。它相当于Android冰山露出水面的部分。最终用户只能看到应用，并乐于对水面下的情况一无所知。但作为开发人员，你必须对此有更深入的了解。

应用是可以占据整个屏幕并与用户进行交互的程序，而服务则隐匿在用户的视线之外，默默地扩展应用框架。本书主要介绍应用的开发，因为我们所编写的大多都是应用。

Android手机和平板电脑在出厂时自带了很多标准系统应用，其中包括：

- 电话拨号程序
- 电子邮件
- 相机
- Web浏览器
- Google Play商店

使用Google Play商店，用户可将新程序下载到自己的手机上运行。这正是你的用武之地。阅读完本书之后，你将能够编写出非常棒的Android应用。

Android应用框架提供了大量可用于创建应用的构件，下面就来对其进行介绍。

2.2　构件

在Android SDK中定义了一些每个开发人员都必须熟悉的对象，其中最重要的是活动、片段、视图、意图、服务和内容提供器。本书后面将通过示例讨论这些对象，下面简要地介绍一下它们。

2.2.1　活动

*活动*是一个用户界面屏幕。应用可以定义一个或多个活动，用于处理程序的不同阶段。正如即将在2.3节进行讨论的，每个活动都负责保存自己的状态，以便能够在应用生命周期的后续阶段恢复这些状态。有关这样的示例请参阅3.2.1节。活动扩展了`Context`类，因此可使用它们来获取应用的全局信息。

2.2.2　片段

*片段*是活动的一个组成部分，通常显示在屏幕上，但并非必须如此。片段是Android 3.0（Honeycomb）引入的，如需针对较旧的Android版本时，可使用兼容库。

在电子邮件程序中，一部分用于显示收到的所有邮件，还有一部分用于显示特定邮件的内容，这两部分都可实现为片段。通过使用片段，能够让应用更轻松地适应不同尺寸的屏幕，如图2-2所示。

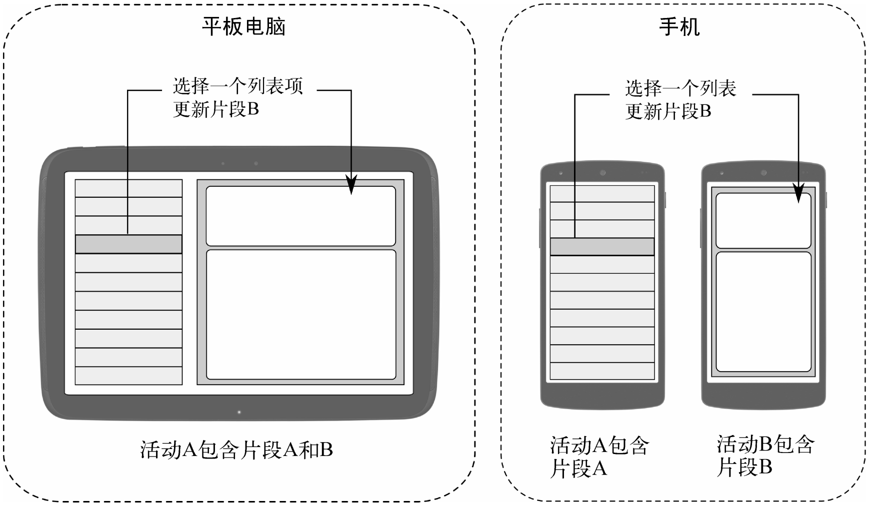


图 2-2

2.2.3 视图

视图是最小的用户界面单元，可以直接包含在活动中，也可以包含在活动的片段中。视图可使用Java代码来创建，但更佳的方式是使用XML布局来定义。每个视图都有一系列的属性，它们决定了视图的功能、行为和外观。

2.2.4 意图

意图是一种行为（如选择照片、给家里打电话、打开舱门）描述机制。在Android中，几乎一切都是通过意图来实现的，这给我们提供了大量替换或重用组件的机会。有关意图的示例，请参阅10.1节。

例如，有一个用于发送电子邮件的意图，如果你的应用需要发送邮件，则可调用这个意图。另外，在编写电子邮件应用时，可以注册一个活动，让它来处理上述意图，从而替换标准邮件程序。这样，当用户再次发送邮件时，就可以使用你编写的程序，而不是标准邮件程序。

2.2.5 服务

服务是在后台运行的任务，无需用户与之直接进行交互，类似于Linux守护程序。就拿音乐播放器来说吧。音乐可能是由活动播放的，但你可能希望它不断播放，即便用户已经切换到了其他程序。因此，实际执行播放的代码应该放在服务中。然后，可能会将另一个活动绑定到该服务，让它负责切换音轨或停止播放。

Android自带了很多内置服务，还有能够方便访问这些服务的API。Google还提供了支持额外功能的可选服务，详情请参阅第12章。

2.2.6　内容提供器

内容提供器是一组数据和用于读取它们的自定义API，这是在应用之间共享全局数据的最佳方式。例如，Google提供了一个包含通讯录的内容提供器，其中所有信息（包括姓名、地址、电话号码等）都可供任何应用使用。有关这方面的示例，请参阅13.5节。

2.2.7　使用资源

资源是本地化的文本字符串、位图或程序需要的其他非代码信息。在编译阶段，所有资源都将被编译到应用中，这有助于国际化和对多种设备的支持。详情请参阅8.3.1节。

你将在项目的res目录中创建和存储资源。Android资源编辑器（aapt）[①]根据资源文件所属的文件夹和格式对其进行处理。例如，对于PNG和JPG格式的位图，应放在目录res/drawable下，而描述布局的XML文件应放在目录res/layout下。可以添加相应的后缀，以指定语言、屏幕朝向、像素密度等。详情请参阅8.3节。

资源编译器对资源进行压缩和打包，再生成一个名为R的类，其中包含可用于在程序中引用这些资源的标识符。这与使用字符串键引用的标准Java资源稍有不同。通过使用标识符来引用资源，能够让Android确保所有的引用都是有效的，同时还避免了存储所有的资源键，从而节省了空间。第3章将通过一个示例演示如何在代码中访问资源。

下面来更详细地介绍Android应用的生命周期，它与桌面应用程序的生命周期稍有不同。

2.3　前台只能有一个应用

在标准Linux或Windows台式机中，可以有很多应用程序同时运行，它们可以同时出现在不同的窗口中。其中一个应用程序拥有键盘焦点，除此之外，所有的应用程序都一样。用户可以轻松地在应用程序之间进行切换并移动窗口（以便能够看到当前执行的操作），以及将不需要的程序关闭。

Android的工作原理则不同。

在Android中只有一个前台应用，它通常占据除状态栏以外的整个屏幕。用户开启手机或平板电脑时，看到的第一个应用为主屏幕，如图2-3所示。

① http://d.android.com/tools/building



图 2-3

用户运行应用时，Android会启动该应用并让它进入前台。在运行该应用时，用户可能还会不断调用其他应用或当前应用中的其他屏幕。系统的活动管理器会将所有这些程序和屏幕都记录到应用栈中。每当用户按“返回”按钮时，都将返回到栈中的前一个屏幕。从用户的角度来看，这很像Web浏览器的历史记录，按“后退”按钮将返回到前一个页面。

2.3.1 进程不等于应用

在内部，每个用户界面屏幕都由一个活动表示（参见2.2.1节）。每个活动都有其生命周期。应用由一个或多个活动以及包含这些活动的Linux进程组成。这听起来相当简单，不是吗？但别高兴得太早，难懂的地方就在后面。

在Android中，应用在其进程终止后也能存活。换句话说，活动的生命周期并不受制于进程的生命周期。进程并不是一次性的活动容器。

2.3.2 活动的生命周期

在生命周期内，Android程序的活动可处于多种状态，如图2-4所示。开发人员并不能控制程序所处的状态，这完全由系统管理。然而，在状态发生变化前，系统会调用方法`onXX()`来通知你。

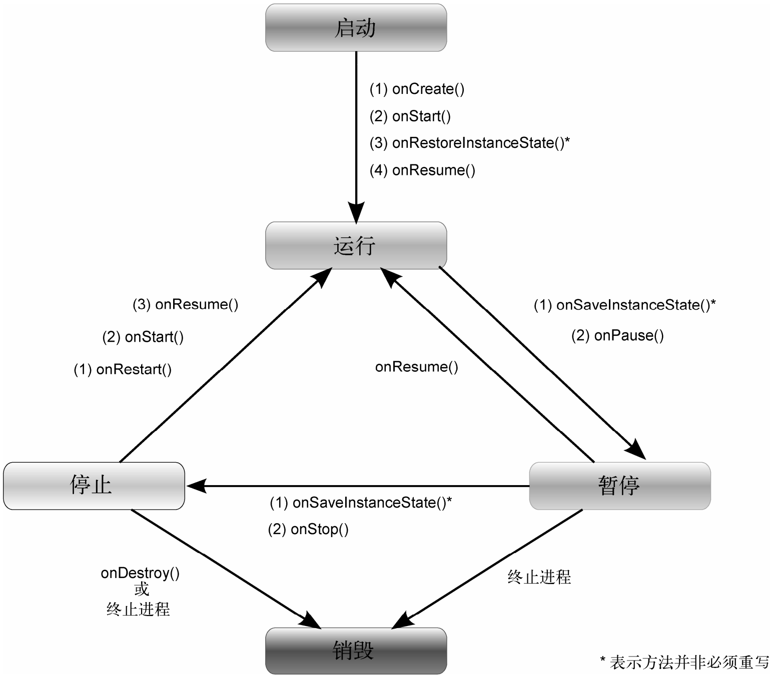


图 2-4

在创建的`Activity`子类时，需要重写如下方法，Android会在合适的时候调用它们。

- `onCreate(Bundle)`：活动首次启动时被调用。可以使用它来执行一次性初始化工作，如创建用户界面。`onCreate()`接受一个参数，这个参数要么为`null`，要么为方法`onSaveInstanceState()`保存的状态信息。
- `onStart()`：表明活动即将显示给用户。
- `onResume()`：活动能够开始与用户交互时被调用。这是启动动画和音乐的理想场所。
- `onPause()`：在活动即将进入后台（通常是由于启动了另一个活动）时被调用。应该在这个方法中保存程序的持久化状态，如正在编辑的数据库记录。
- `onStop()`：在活动对用户不再可见且暂时不需要该活动时被调用。如果内存紧张，`onStop()`可能不会被调用（系统可能会直接终止进程）。
- `onRestart()`：如果该方法被调用，就表明原本处于停止状态的活动重新显示到了屏幕上。
- `onDestroy()`：在活动销毁前被调用。如果内存紧张，`onDestroy()`可能不会被调用（系统可能会直接终止进程）。
- `onSaveInstanceState(Bundle)`：Android调用这个方法让活动保存其特有的状态，如光标在文本框中的位置。通常不需要重写这个方法，因为默认实现会自动保存所有用户界面控件的状态。

- `onRestoreInstanceState(Bundle)`：根据`onSaveInstanceState()`方法保存的状态重新初始化活动时被调用，其默认实现会恢复用户界面的状态。

不在前台运行的活动可能会被停止。另外，包含此类活动的Linux进程也可能随时被终止，以便为新活动腾出空间。这种情况屡见不鲜。因此在设计应用时，一开始就必须考虑到这一点。这很重要。在某些情况下，`onPause()`可能是最后一个被调用的活动方法，因此对于要保留到下次使用的任何数据，都必须在这里进行保存。

从Android 3.0（Honeycomb）起，Google在应用生命周期的故事中引入了另一个情节——片段。

2.3.3 使用片段简化工作

片段是应用的组成部分，它们包含在活动中（参见2.2.2节），其生命周期与活动很像。事实上，片段的很多生命周期方法都是由活动的方法调用的（例如，`Fragment.onResume()`间接地由`Activity.onResume()`调用）。详情请参阅图2-5。

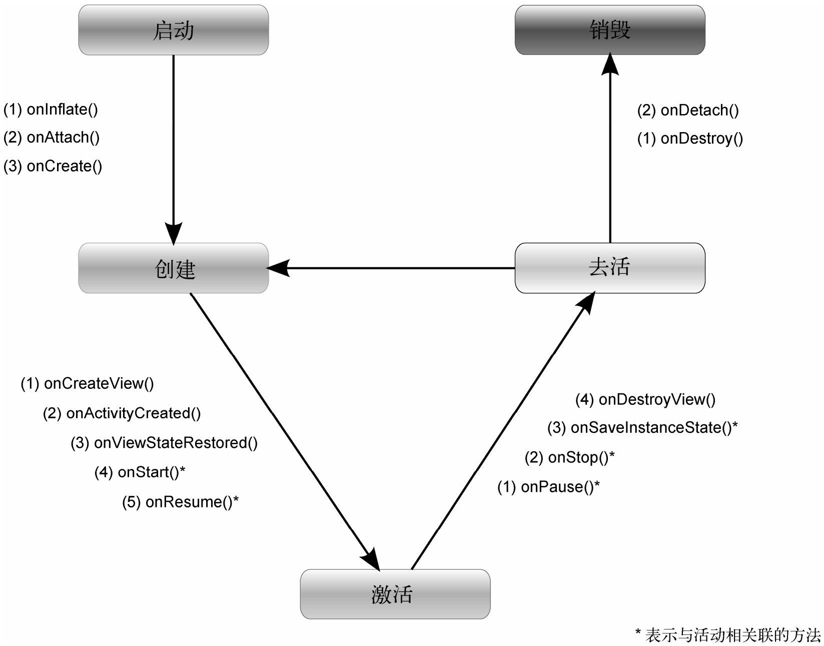


图 2-5

不存在包含它们的活动时，片段依然可以存在。例如，如果你在应用运行时旋转屏幕，活动通常会被销毁并重新创建，以适应新的屏幕朝向，但片段通常会被保留。这样能够在朝向切换期间保留网络连接等重量级对象。

2.4　安全保障

前面说过，每个应用都运行在自己的Linux进程中。硬件禁止一个进程去访问另一个进程的内存。另外，每个应用都被分配了一个独特的用户ID。只要设备没有被root（以更高的权限运行应用），一个应用创建的文件就不能被其他应用读写。

另外，限制了对一些关键操作的访问。要使用这些操作，必须在文件AndroidManifest.xml中请求相应的权限。在应用安装时，包管理器将根据证书授予或不授予请求的权限，并在必要时询问用户。下面是一些最常见的权限。

- INTERNET：访问Internet。
- READ_CONTACTS：读取但不写入用户的联系人数据。
- WRITE_CONTACTS：写入但不读取用户的联系人数据。
- RECEIVE_SMS：监视收到的短信。
- ACCESS_COARSE_LOCATION：使用粗糙的位置提供器，如基站或Wi-Fi。
- ACCESS_FINE_LOCATION：使用更精确的位置提供器，如GPS。

例如，要监视收到的短信，需要在清单文件中包含如下内容：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.google.android.app.myapp" >
   <uses-permission android:name="android.permission.RECEIVE_SMS" />
</manifest>
```

Android甚至可以限制访问系统的特定部分。通过在AndroidManifest.xml中使用XML标签，可指定谁可启动活动、启动或绑定服务、向接收器广播意图以及访问内容提供器中的数据。这种高级控制不在本书的讨论范围内，要更深入地了解，可参阅有关Android安全模型的在线帮助[①]。

2.5　快速阅读指南

本书余下的篇幅将使用本章介绍的所有概念。第3章将使用活动和生命周期方法来定义一个示例应用，第6章将探索多媒体编码解码器，而第13章将介绍内容提供器。

① http://d.android.com/training/articles/security-tips.html

第二部分

开发一个游戏

第 3 章

开局走法

在第1章，我们使用Android Studio编写了一个简单的“Hello, Android”程序。这在几分钟内就完成了，但功能有限。在第二部分，我们将创建一个有趣得多的应用——终极版井字游戏（Ultimate Tic-Tac-Toe）。

通过不断给这个程序添加功能，你将学习Android编程的很多方面。这些知识将有助于你编写自己的程序——游戏、商业应用以及你能想象得到的任何程序。我们将从用户界面着手。

在本书中，每个示例都被完整地呈现了出来，以便能够让你在阅读的同时在Android Studio中跟着做。你可以从本书配套网站[①]下载这些示例，这样可以节省大量的代码输入时间。如果你阅读的是电子版，那么可以单击代码清单前面的文件名，直接下载该文件。

3.1　创建井字游戏示例

人人都会玩井字游戏：一个3×3的棋盘，最初是空的，两个玩家轮流在空格中填写X或O，先将自己的3个棋子连成线者获胜。如果棋盘填满后还没人将3个棋子连成线，就视为平局。

终极版井字游戏[②]与此类似，只不过每格都内嵌了一个井字，玩家不是通过下子来占据各个格子，而是需要玩赢格子内的井字游戏才能占据它，如图3-1所示。

终极版井字游戏有很多变体，这里采用这样的变体：在大格包含的井字游戏为平局时，算双方都赢，其规则与流行的Android和iOS游戏“井字策略（Tic Tactics）”[③]类似。（需要指出的是，我与“井字策略”游戏的开发商一点儿关系都没有，但要不是因为我花了太多时间在玩这个游戏上，也许本书在一年前就付梓了。）

为了新建一个程序，选择菜单File➤New Project并输入如下信息。

- 应用名：Tic Tac Toe

---

① http://pragprog.com/book/eband4

② http://mathwithbaddrawings.com/2013/06/16/ultimate-tic-tac-toe

③ http://www.hiddenvariable.com/tictactics/

- 公司域名：example.org
- 尺寸：Phone and Tablet
- 最低SDK：API 16: Android 4.1 (Jelly Bean)
- 添加活动：Blank Activity
- 活动名：MainActivity
- 布局名：activity_main
- 标题：UT3

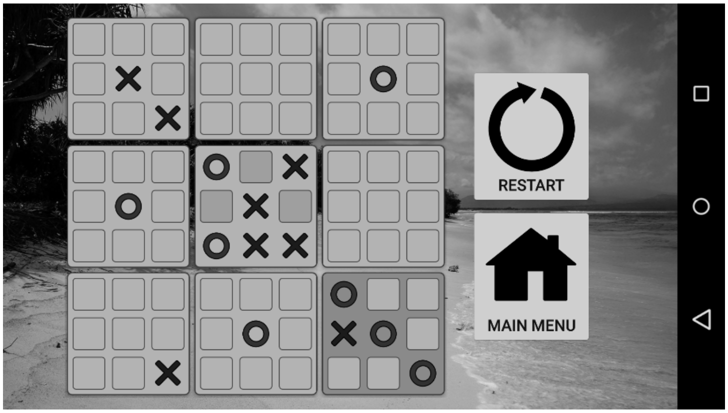


图3-1　井字游戏遇上《盗梦空间》

可将目录res/menu删除，因为这个项目不需要它。

在编写这个程序的实际代码前，需要设计起始屏幕。为此，我们将编辑一个XML布局文件。

3.2　使用 XML 进行设计

在Android中，定义用户界面的方式有两种：使用代码和使用XML。通常推荐使用XML，因为相比于Java代码，使用XML来创建界面以及设置界面样式会更容易。

Android Studio包含一个可视化布局编辑器，让你能够使用鼠标将按钮和列表等控件拖放到画布上，然后调整它们的位置，最终创建出漂亮的用户界面。你应该好好研究一下该编辑器的用法并多多加以使用。不过本书只负责介绍如何处理它创建的XML文本，以便让你更深入地了解相关工作原理。

3.2.1　创建主屏幕

我们希望这个程序启动时显示如下选项：接着玩游戏、开始新游戏以及显示有关该程序的信息。图3-2显示了我们要设计的屏幕。



图　3-2

为创建这个屏幕，需要编辑文件activity_main.xml。为此，进入Android Studio的项目（Project）窗口，并双击文件夹res/layout中的文件名activity_main.xml。请注意，项目窗口有3种模式：Project、Packages和Android。要切换模式，可使用窗口名旁边的下拉列表。

如果没有看到XML，请单击编辑器窗口底部的Text标签。将文本修改成如下内容。

ticTacToev1/src/main/res/layout/activity_main.xml

```
<FrameLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:clipChildren="false"
   tools:context=".TicTacToeActivity">

   <fragment
      android:id="@+id/main_fragment"
      class="org.example.tictactoe.MainFragment"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center"
      tools:layout="@layout/fragment_main"/>
</FrameLayout>
```

这里定义了一个`FrameLayout`视图，它覆盖了整个屏幕，其中有一个浮动的片段（该片段为类`MainFragment`的实例）。如果要让代码正确地对齐，可选择菜单Code➤Reformat Code，也可使用快捷键（在Windows中为Ctrl+Alt+L）。

如果你是手动将这些代码输入到Android Studio中的，将发现有些文本是红色的，且预览窗

口中显示了错误消息（如果打开了该窗口）。这是因为其中有两个部分还没有定义，稍后将添加它们。

FrameLayout是Android所支持的多种布局视图中的一种。这些视图会对它们所包含的视图进行排列，以创建出所需的效果。布局有很多种，你还可以创建自定义布局。下面总结了最常见的布局类型，你在程序中很可能会用到。

- FrameLayout：以堆叠方式显示一个或多个子视图。
- GridLayout：将子视图按行和列排列。
- LinearLayout：将所有子视图排列成一行或一列。
- RelativeLayout：一种灵活的布局，以相对于其他视图的方式排列视图。

3

有关Android布局的完整列表，以及对每种布局可设置的选项的详情，请参阅在线文档[①]。

由于只有一个子视图，因此使用的布局不会影响该视图的外观。由于FrameLayout最简单且效率最高，因此我们选择使用这种布局。

1. FrameLayout标签的属性

每个XML标签都有控制器功能的属性，下面更深入地介绍前面给FrameLayout标签设置的属性。

- xmlns:android="http://schemas.android.com/apk/res/android"：定义命名空间[②]android，以便能够在后面使用包含android:的属性名。
- xmlns:tools="http://schemas.android.com/tools"：定义命名空间tools。
- android:layout_width="match_parent"：将视图设置为与父视图等宽。由于是顶级元素，这意味着它将与屏幕等宽。该属性可设置为match_parent、wrap_content或绝对宽度值。
- android:layout_height="match_parent"：将视图设置为与父视图（屏幕）等高。对于每个视图，都必须设置宽度和高度。
- tools:context=".TicTacToeActivity"：指出该布局文件是供TicTacToeActivity类使用的。不同于其他属性，该属性是供可视化编辑器使用的，而不是在运行阶段使用的。

2. fragment标签的属性

下面是前面为fragment标签设置的所有属性的含义。

- android:id="@+id/fragment_main"：定义新资源标识符fragment_main，在代码或其他XML属性中使用。@+表示定义新内容，@表示引用已在其他地方定义过的内容。
- class="org.example.tictactoe.MainFragment"：让Android知道这个片段是MainFr

① http://d.android.com/guide/topics/ui/declaring-layout.html

② http://en.wikipedia.org/wiki/XML_namespace

agment类的一个实例。换句话说，Android在根据XML创建该片段时，将创建一个MainFragment实例。这个类将在3.3.2节定义。

- android:layout_width="wrap_content"：将片段设置为与其包含的内容等宽。
- android:layout_height="wrap_content"：将片段设置为与其包含的内容等高。
- android:layout_gravity="center"：将片段在其父视图中居中设置。该属性的可能取值包括top、bottom、left、right、center和fill等。可将该属性设置为多个用竖线连接的值，如top|right。
- tools:layout="@layout/fragment_main"：引用另一个XML文件，该XML文件定义了片段的内容。

3.2.2　创建主片段

主片段包含3个按钮，分别用于继续游戏、开始新游戏以及显示有关游戏的信息。我们将在fragment_main.xml中定义它们。

要创建这个文件，可单击文件activity_main.xml，然后依次按Ctrl+C（复制）和Ctrl+V（粘贴）。在系统提示指定新的文件名时，输入fragment_main.xml（还有其他创建文件的方式，如使用向导，但在我看来，这种方式是最快的）。

将fragment_main.xml的内容替换为如下文本。

ticTacToev1/src/main/res/layout/fragment_main.xml

```
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:background="@drawable/menu_background"
   android:elevation="@dimen/elevation_high"
   android:orientation="vertical"
   android:padding="@dimen/menu_padding"
   tools:context=".TicTacToeActivity">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginBottom="@dimen/menu_space"
      android:text="@string/long_app_name"
      android:textAppearance="?android:textAppearanceLarge"
      android:textSize="@dimen/menu_text_size"/>

   <Button
      android:id="@+id/continue_button"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_margin="@dimen/menu_button_margin"
```

```
        android:padding="@dimen/menu_button_padding"
        android:text="@string/continue_label"/>

    <Button
        android:id="@+id/new_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="@dimen/menu_button_margin"
        android:padding="@dimen/menu_button_padding"
        android:text="@string/new_game_label"/>

    <Button
        android:id="@+id/about_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="@dimen/menu_button_margin"
        android:padding="@dimen/menu_button_padding"
        android:text="@string/about_label"/>
</LinearLayout>
```


其中很多行都显示为红色。不用担心，本章后面将修复这些问题。

在这里，要将4个元素排成一列，因此使用了一个`LinearLayout`，并将其`android:orientation`属性设置为`vertical`。在这个线性布局内部，定义了一个文本视图和3个按钮。

在Android开发中，文本视图可能是最常见的用户界面元素了。下面列出了其他一些常见的界面元素。

- Button：按钮控件。
- CheckBox：有两种可能状态的按钮，用于多选列表中。
- EditText：可编辑的文本视图。
- ImageButton：显示图像而不是文本的按钮。
- ListView：在可固定的垂直列表中显示一系列列表项。
- RadioButton：有两种可能状态的按钮，用于单选列表中。
- VideoView：显示视频文件。
- WebView：显示网页。

要获悉完整的用户界面元素列表，请参阅有关`View`类的文档[①]。

1. LinearLayout标签的属性

在前面为`LinearLayout`标签设置的属性中，尚未介绍的属性如下。

- `android:background="@drawable/menu_background"`：设置整个视图的背景drawable。

① http://d.android.com/reference/android/view/View.html

drawable可以是颜色、图形或XML文件定义的复合对象，稍后将对其进行更详细的介绍。

- ❑ `android:elevation="@dimen/elevation_high"`：令视图比画布稍高。在Android 5.0（Lollipop）和更高版本中，这将导致在视图下方绘制阴影。如果运行程序的设备使用的Android版本较旧，就像其他无法识别的属性一样，忽略该属性。这里没有使用数字来明确指定视图要比画布高多少，而是引用了一个将在另一个XML文件中定义的尺寸。
- ❑ `android:orientation="vertical"`：指定布局的方向，可能的取值为`vertical`（将子视图排成一列）和`horizontal`（将子视图排成一行）。如果要将子视图排成多行、多列，必须使用其他布局，如`GridLayout`。
- ❑ `android:padding="@dimen/menu_padding"`：让Android在视图内部留出少量的空间。如果要在视图外部留出空间，可使用属性`margin`。

间接值（如`@dimen/menu_padding`）是很有用的，但有时可能令人迷惑。此时Android Studio可提供帮助，它会在可能的情况下在编辑器中显示最终解析得到的值。间接值被定义后，可将鼠标指向它或单击它以显示原始引用，再按住Ctrl并单击引用以查看其定义。

2. TextView标签的属性

前面为`TextView`标签设置了如下属性。

- ❑ `android:layout_marginBottom="@dimen/menu_space"`：在文本视图和按钮之间留出一定的空间。
- ❑ `android:text="@string/long_app_name"`：指定要显示的文本。这里引用了将在文件strings.xml中定义的`long_app_name`。
- ❑ `android:textAppearance="?android:textAppearanceLarge"`：让文本字体比常规状态更大、更粗。?表示引用了当前主题中定义的一个常量。主题定义了数百个常量，用于控制应用中每个视图的外观和行为。更详细的信息请参阅3.5.5节。
- ❑ `android:textSize="@dimen/menu_text_size"`：即便设置了属性`textAppearance`，文本看起来还不够大，因此这里用硬编码的方式指定了更大的字号。

3. Button标签的属性

为`Button`标签设置如下属性。

- ❑ `android:layout_margin="@dimen/menu_button_margin"`：在按钮周围留出一些额外的空间。
- ❑ `android:padding="@dimen/menu_button_padding"`：在按钮内部留出一些额外的空间。
- ❑ `android:text="@string/continue_label"`：指定按钮显示的文本。这里再次引用了将在文件strings.xml中定义的值。

对于前面所有以@打头的值，都将在3.5节中定义。下面先来编写与XML协同工作的Java代码。

3.3 编写代码

定义主屏幕和片段的布局后，需要编写一些Java代码让它们显示出来。先从该应用的主活动——`MainActivity`类开始。

3.3.1 定义主活动

在org.example.tictactoe包中，文件夹java下应该有一个名为MainActivity.java的文件，它是在我们创建项目时由Android Studio创建的。现在，请打开它。为此，可在项目窗口中双击它；也可选择菜单Navigate➤File或按其快捷键（在Windows中Ctrl+Shift+N），然后输入文件名。接下来，将其中的所有代码替换为如下内容。

ticTacToev1/src/main/java/org/example/tictactoe/MainActivity.java

```
Line 1 package org.example.tictactoe;
     -
     - import android.app.Activity;
     - import android.os.Bundle;
     5
     - public class MainActivity extends Activity {
     -
     -    @Override
     -    protected void onCreate(Bundle savedInstanceState) {
    10       super.onCreate(savedInstanceState);
     -       setContentView(R.layout.activity_main);
     -    }
     - }
```

这是本书第一次介绍Java代码，因此将逐行介绍每行代码的功能。

第1行定义了包名。为方便访问以及避免名称冲突，将Java源文件收集到了包中。包名必须与目录名匹配。

第3行和第4行告诉编译器，我们将使用android.app包中的`Activity`类以及android.os包中的`Bundle`类。这两个包都是Android框架提供的标准包。

从第6行开始，是`MainActivity`类的定义，它是`Activity`类的子类。关于活动，已经在第2章中讨论过。

从第9行开始是方法`onCreate()`，它是活动生命周期的一部分，在活动创建时被调用。`@Override`指出这个方法最初是在`Activity`类中定义的，但我们将给它提供新的定义。在新定义中，首先会调用旧定义，如第10行所示。

第11行很重要，它使用activity_main.xml定义的XML布局填充活动的内容，用来演示如何使用前面声明的XML。

最后，由最后两行来指定代码块的结束位置。所有的代码都必须正确地嵌套，否则编译器将报错。

下面来定义主活动使用的片段。

3.3.2　定义主活动使用的片段

在3.2.1节中，我们使用`org.example.tictactoe.MainFragment`类创建了一个片段。下面来定义这个类。

首先创建文件MainFragment.java。为此，找到文件MainActivity.java并复制它，将复制的文件命名为MainFragment.java，并在该文件打开后，将其所有代码替换为如下内容。

ticTacToev1/src/main/java/org/example/tictactoe/MainFragment.java

```
package org.example.tictactoe;

import android.app.AlertDialog;
import android.app.Fragment;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class MainFragment extends Fragment {

   private AlertDialog mDialog;

   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                            Bundle savedInstanceState) {
      View rootView = inflater.inflate(R.layout.fragment_main, container, false);
      // 在这里添加处理按钮的代码……
      return rootView;
   }
}
```

这个类的结构与`MainActivity`类相似，但片段的启动方法为`onCreateView()`。这个方法接受3个参数：一个`inflater`对象（可用于将XML转换为视图）、一个指向父容器的引用以及一些保存的状态。我们不需要保存的状态，而只调用方法`inflater.inflate()`，并传入`R.layout.fragment_main`（它指向前面定义的fragment_main.xml），再返回该方法创建的视图。

这个主片段包含3个按钮，分别用于继续游戏、开始新游戏以及查看有关游戏的信息。下面，首先来实现这3项功能中最简单的一项。

3.4 添加 About 框

很多应用都包含一个这样的按钮，即用户通过它可以获悉有关应用的更详细信息或帮助。你可能希望在用户点击该按钮时显示参与程序开发的所有人员、使用的开源包或其他法律信息。在这个示例中，我们要在用户点击About按钮时显示两个段落，用来指出这个游戏的玩法，如图3-3所示。

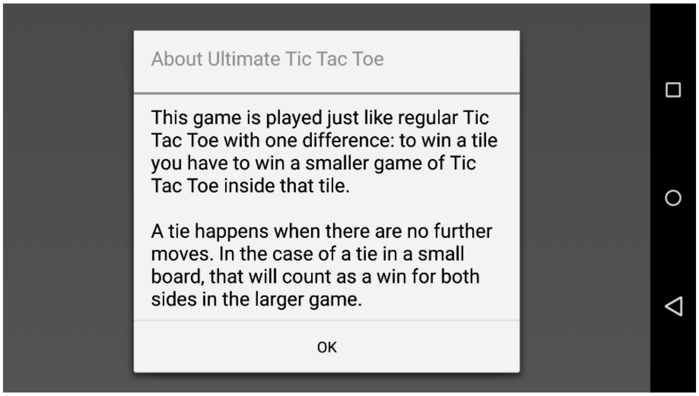


图 3-3

为此，需要在片段的方法`onCreateView()`中添加几行代码，即在返回`rootView`的代码前面，添加如下代码。

ticTacToev1/src/main/java/org/example/tictactoe/MainFragment.java

```
// 处理按钮的代码……
View aboutButton = rootView.findViewById(R.id.about_button);
aboutButton.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {
      AlertDialog.Builder builder =
            new AlertDialog.Builder(getActivity());
      builder.setTitle(R.string.about_title);
      builder.setMessage(R.string.about_text);
      builder.setCancelable(false);
      builder.setPositiveButton(R.string.ok_label,
            new DialogInterface.OnClickListener() {
               @Override
               public void onClick(DialogInterface dialogInterface,
                                    int i) {
                  // 这个方法为空
               }
            });
      mDialog = builder.show();
   }
});
```

第2行负责获取片段视图中的About按钮，然后由第3行为它设置点击监听器。用户轻按该按钮时，将调用监听器的方法`onClick()`（第5行）。

第6行用于创建一个新的`AlertDialog.Builder`实例，并将当前活动作为参数传入。接下来，从第8行开始，设置对话框的标题和消息内容。第10行调用了`setCancelable()`，使得Android不会在用户轻按对话框外面时关闭它。第11行在对话框中添加了一个OK按钮，但它没有任何用途。

定义完对话框后，第19行负责将其显示出来。

还有一项工作要做。我们要在包含该片段的活动暂停（如启动了另一个应用）时，将About框关闭。为此，应在`MainFragment`类中添加如下`onPause()`方法。

ticTacToev1/src/main/java/org/example/tictactoe/MainFragment.java

```
@Override
public void onPause() {
   super.onPause();

   // 如果About对话框未关闭，就将其关闭
   if (mDialog != null)
      mDialog.dismiss();
}
```

至此，`MainFragment`包含了两个方法：`onCreateView()`和`onPause()`。如果编译器报错，请确保所有的大括号都匹配。如果依然有问题，请从本书配套网站[1]下载完整的源代码文件。

3.5　定义资源

几乎所有的资源都是使用XML定义的（参见2.2.7节）。在编译阶段，对XML进行了预编译。这样，程序在运行时不会因为分析XML而降低性能。

3.5.1　字符串

在程序代码和布局中，不以硬编码的方式指定文本字符串，而是将所有的文本字符串都存储在一个地方——文件夹res/values中的资源文件strings.xml。这样，在需要将应用推向外国市场时，翻译这些字符串的工作将容易得多。在程序Tic-Tac-Toe中，当前所使用的字符串如下：

ticTacToev1/src/main/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">UT3</string>
    <string name="long_app_name">Ultimate Tic Tac Toe</string>
```

[1] http://pragprog.com/book/eband4

```
    <string name="action_settings">Settings</string>
    <string name="continue_label">Continue</string>
    <string name="new_game_label">New Game</string>
    <string name="about_label">About</string>
    <string name="ok_label">OK</string>
    <string name="about_title">About Ultimate Tic Tac Toe</string>
    <string name="about_text">\
This game is played just like regular Tic Tac Toe with one difference: to
win a tile you have to win a smaller game of Tic Tac Toe inside that tile.\n\
\n\
A tie happens when there are no further moves. In the case of a tie in a small
board, that will count as a win for both sides in the larger game.
</string>
</resources>
```


每项资源都有名称和值。在Java代码或XML中，将名称用作id来引用资源。例如，在Java代码中，`R.string.app_name`为资源`app_name`的id。要获取实际字符串，必须调用另一个函数（如`Activity.getString()`），并传入相应的id。在XML中，情况更简单些，只需使用`@`，并在其后面指定资源类型和名称即可，如`@string/app_name`。

在8.3.1节，将介绍如何指定替代资源。

3.5.2 尺寸

尺寸资源可用于任何需要指定长度的地方。所有尺寸资源都放在一个尺寸文件（文件夹res/values下的文件dimens.xml）中，这样有助于在不修改代码的情况下支持不同尺寸的Android设备。到目前为止，我们的应用使用了如下尺寸。

ticTacToev1/src/main/res/values/dimens.xml

```
<resources>
   <dimen name="elevation_high">8dp</dimen>
   <dimen name="stroke_width">1dp</dimen>
   <dimen name="corner_radius">4dp</dimen>
   <dimen name="menu_padding">10dp</dimen>
   <dimen name="menu_space">10dp</dimen>
   <dimen name="menu_text_size">32sp</dimen>
   <dimen name="menu_button_margin">4dp</dimen>
   <dimen name="menu_button_padding">10dp</dimen>
</resources>
```

有关dp和sp的含义，请参阅3.5.6节。

你可能会看到两个版本的dimens.xml文件：常规版本和使用w820dp标记的版本。目前只使用了常规版本，另一个版本用于宽屏设备的替代资源。有关替代资源，将在8.3.1节讨论。

3.5.3 drawable

drawable指的是可在屏幕上绘制的任何图形对象。位图是最简单的drawable，通常以PNG或

JPG格式存储。在主屏幕上，应用的启动图标就是位图。

还可以使用XML来创建drawable。下面是用作主屏幕中选项的背景drawable的定义，它被放置在文件夹res/drawable中。

ticTacToev1/src/main/res/drawable/menu_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:shape="rectangle">
   <stroke
      android:width="@dimen/stroke_width"
      android:color="@color/border_color"/>
   <solid android:color="@color/field_color"/>
   <corners android:radius="@dimen/corner_radius"/>
</shape>
```

该XML文件定义了一个矩形形状，该形状带圆角并使用纯色填充。注意到它引用了其他资源，以指定描边的宽度和颜色、填充色以及圆角半径。

这也可以使用位图来实现，但使用XML具有如下优点：图形是基于矢量的，支持任意分辨率。这意味着不管将该背景放大到多大，它都是清晰的，不会导致像素化。

下面是最常见的drawable类型。

- 位图文件：PNG、JPG或GIF格式的图片。PNG位图可以是半透明的。
- 九宫格（Nine-patch）文件：包含可拉伸区域的PNG图片，支持根据内容调整大小。
- 图层列表（Layer list）：一系列按顺序绘制的drawable。
- 状态列表（State list）：一系列drawable，根据状态显示不同的drawable。
- 等级列表（Level list）：一系列drawable，根据等级值显示不同的drawable。
- 形状：由线条和颜色组成的几何形状。

有关drawable类型及其属性的完整列表，请参阅在线文档[①]。

3.5.4　颜色

下面是前面的背景资源使用的颜色的定义。

ticTacToev1/src/main/res/values/colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="field_color">#b5cee0</color>
   <color name="border_color">#7f7f7f</color>
</resources>
```

① http://d.android.com/guide/topics/resources/drawable-resource.html

在Android中，颜色是以#RRGGBB或#AARRGGBB的形式指定的。其中，RR、GG、BB分别以十六进制的方式指定了红色、绿色和蓝色组分，AA为alpha组分。这些十六进制数字的取值范围为00（0）~FF（255）。例如，#FF0000表示纯红色，而#FFFFFF表示白色。

alpha组分是可选的，表示颜色的透明度，取值范围为0（完全透明）~255（完全不透明）。如果没有设置alpha组分，颜色将是完全不透明的。

3.5.5 样式和主题

样式和主题在决定应用的外观方面扮演着重要角色。主题是一系列样式，而样式是一系列控制外观和行为的值。如果你打开文件AndroidManifest.xml，将看到下面一行。

```
android:theme="@style/AppTheme"
```

它引用了styles.xml中所定义的样式`AppTheme`。请按住Ctrl键并单击AppTheme以打开文件styles.xml，然后将该文件做如下修改。

ticTacToev1/src/main/res/values/styles.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <!-- 应用的基本主题 -->
   <style name="App Theme"
          parent="android:Theme.Holo.Light.NoActionBar.Fullscreen">
      <!-- 在这里定制主题 -->
   </style>
</resources>
```

主题定义了数百个值，因此通常会使用定义好的主题，并修改其中的几个值。为指定现成的主题，可使用属性`parent`。在这里，我们使用了顶部没有标题栏的主题，以便让活动覆盖整个屏幕。

要查看所有可用的主题，可在Android Studio编辑器中将光标放在`android:Theme.`后面，再按Ctrl+空格键。应用运行后，你可能希望回过头来使用不同的主题，看看它们如何影响应用。

3.5.6 dp和sp

以前，程序员设计计算机界面时总是以像素为单位。例如，可能将文本框的宽度设置为300像素，将列间距设置为5像素，将图标大小指定为16×16像素。这样做之后带来的问题是，运行程序时，设备屏幕的每英寸点数（dpi）越多，界面越小。dpi高到一定程度后，将很难看清界面。

为帮助解决这种问题，可使用独立于分辨率的度量单位。Android支持如下度量单位。

- px（像素）：屏幕上的点。
- in（英寸）：用标尺量度的尺寸。
- mm（毫米）：用标尺量度的尺寸。

- pt（磅）：1/72英寸。
- dp（密度无关像素）：基于屏幕密度的抽象单位。在160 dpi屏幕上，1 dp = 1 px。
- dip：含义与dp相同。
- sp（比例无关像素）：类似于dp，可根据用户的字号首选项设置进行缩放。

为了让界面能够适应当前和未来的屏幕类型，建议在指定文本大小时以sp为单位，而指定其他内容的大小时都以dp为单位。在任何时候，都不要以像素（px）为单位。

3.5.7　运行游戏

万事俱备，现在可以运行这个游戏了。为此，可选择菜单Run➤Run 'app'、单击工具栏上的Run按钮或使用快捷键（在Windows中为Shift+F10）。与运行“Hello, Android”程序时一样，这里会要求你选择设备。可以选择模拟器，也可以选择通过USB连接到计算机的实际设备。

如果一切顺利，你将看到类似于图3-2所示的屏幕。如果你点击按钮Continue或New Game，什么都不会发生（这些功能将在下一章添加）；而如果你点击About按钮，将出现About对话框，请点击OK按钮将其关闭。

小乔爱问：

如何退出呢？

你可能注意到了，这个游戏没有退出按钮。这是因为根本就不需要这种按钮。如果添加这种按钮，反倒有悖于 Android 设计指南[1]。要退出游戏，只需按返回按钮或主屏幕按钮，也可按近期使用的应用（Recent Apps）按钮，并从列表中选择其他应用。

你可能担心这会让程序继续运行。根本没必要。在其他程序需要资源时，Android 将终止该程序，并收回它占用的资源。要强行终止程序，可按近期使用的应用（Recent Apps）按钮，并轻扫它。这不仅会将应用从列表中删除，还将终止它并收回它占用的资源。在开发阶段，当运行了多个存在 bug 的程序时，这是很有用的。但是，在大多数情况下，完全没有必要这样做。

1. http://d.android.com/design

3.6　调试

如果你有编程经验，肯定知道在大多数时候，你并不能如愿以偿。在这种情况下，该怎么办呢？所幸，在其他平台上使用的调试手法也适用于Android，包括将消息打印到日志以及在调试

器中以步进方式执行程序。

3.6.1 使用日志消息进行调试

Log类提供了多个静态方法，可用于将各种等级的消息打印到Android系统日志。

- Log.e()：错误。
- Log.w()：警告。
- Log.i()：提示。
- Log.d()：调试。
- Log.v()：详情（Verbose）。
- Log.wtf()：致命错误。

用户永远都看不到系统日志，但开发人员可以两种方式查看它。在Android Studio中，程序运行时，LogCat视图将出现在窗口底部，如图3-4所示。为减少显示的内容，可将日志等级设置为除Verbose外的其他值，也可在筛选文本框中输入一个字符串。

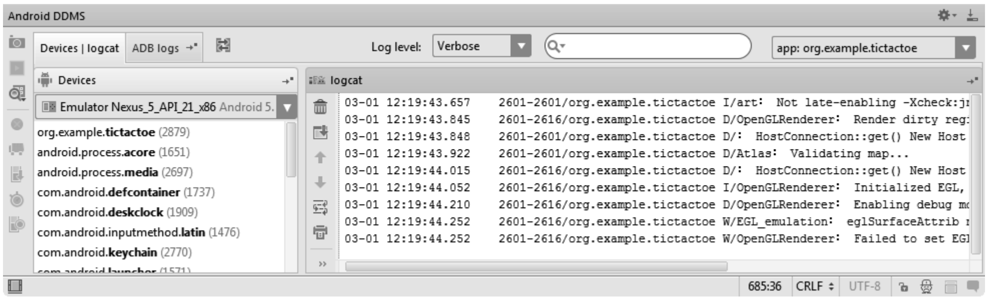


图 3-4

请在程序中输入类似如下的代码行。

```
Log.d("UT3", "Got to point A");
```

这样，下次运行程序时将在日志中看到它们。

如果你使用的不是Android Studio，可采用如下方法查看这种输出：切换到SDK安装目录下的文件夹platform-tools，并运行命令`adb logcat`[①]。你可能会在一个独立的窗口中运行这个命令，并在模拟器运行或设备连接期间始终打开这个窗口。这不会影响其他任何监视工作。

在开发期间，要特别强调Android日志的重要性。每当遇到意外的错误时，都请先查看日志。十有八九你都可以通过它获得足够的信息，进而将问题排除，而无需使用重武器——调试器。

① http://d.android.com/tools/help/adb.html

3.6.2　使用调试器进行调试

Android Studio提供了一个调试器，它是排除Android程序中的问题的终极方式。要启动调试器，请选择菜单Run➤Debug（而不是Run➤Run），也可以单击工具栏中的“调试”按钮或使用快捷键（Windows中为Shift+F9）。

在启用了调试器的情况下运行时，应用在发生异常或到达断点时都将停止运行。断点类似于禁行标志，可在代码中放置它们。为此，需要在Java编辑器中，单击要暂停的代码行左边的gutter，单击处将出现一个红色图标。下次程序运行到相应代码行时将暂停执行。

暂停执行后，屏幕底部将出现调试窗口，如图3-5所示。其中显示了当前位置以及所有变量的值。

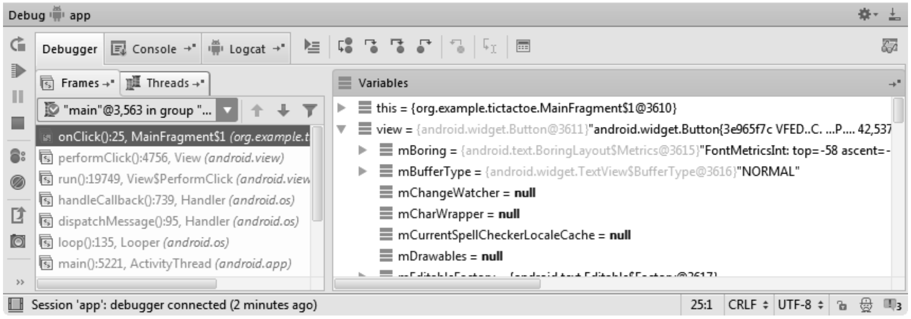


图　3-5

在这个窗口中，你可以检查变量和类成员、查看到达当前位置的调用序列（追踪）、继续往下执行到下一个断点或以每次一行的方式执行程序。

3.6.3　测试

除即兴测试（即测试程序能否正常运行）外，强烈建议尽可能使用自动测试。可以运行各种测试，最常见的测试如下。

- 单元测试[①]：检查程序的低级功能。单元测试基于JUnit框架[②]，Android Studio和gradle都为单元测试提供了支持。
- 用户界面测试[③]：检查用户界面能否正常运行。通常使用脚本语言来与用户界面进行一系列交互。

① http://d.android.com/tools/testing/testing_android.html
② http://junit.org
③ http://d.android.com/tools/testing/testing_ui.html

- 猴子测试[1]：使用随机输入对程序狂轰乱炸，试图破坏它。这犹如数千只猴子排成队，依次随机地点击屏幕上的按钮，直到程序崩溃。

虽然进行自动测试不容易，但它们对专业项目来说必不可少。全面讨论自动测试超出了本书的范围，网上有大量有关这方面的教程和指南。

3.7 快速阅读指南

本章介绍的内容很多。你白手起家，使用了布局文件来组织用户界面，并使用Android资源来提供文本、颜色等。你还创建了表示活动和片段的Java类，并添加了按钮和文本框等控件。

Android是个复杂的系统，但无需全面了解它就可开始编程。需要帮助时，可参阅数百页在线参考资料，它们更深入地介绍了所有的类和方法。

要查看在线文档，可打开Android SDK安装目录下的子目录docs，也可以通过浏览器访问http://d.android.com。当然，每当你遇到困难时，都可以访问本书配套的在线论坛[2]。

其他读者和作者将很乐意帮你摆脱困境。另一个不错的资源是Stack Overflow[3]，也可以使用Google进行搜索。

信不信由你，即便在Google和网络面世前，人们也能编写程序。但现在如果没有网络，做什么事情都难上加难。这也是提倡购买大量图书的另一个原因！

① http://d.android.com/tools/help/monkey.html
② https://forums.pragprog.com/forums/231
③ http://stackoverflow.com/questions/tagged/android

第 4 章 定义游戏界面

前一章创建了一个终极版井字游戏，并为之实现了开始屏幕。还有哪些没有实现呢？游戏部分。我们将分两章完成这项任务。在本章中，我们将创建该游戏的用户界面。你将能够与另一个人交替地使用你的手机或平板电脑来一起玩这款游戏。在下一章，我们将让这款游戏能够思考并下棋，让你能够一个人玩。

4.1 棋盘

为创建棋盘，我们将从最小的部分着手，一步一步处理越来越大的部分。你也可以根据你的思维方式，从顶级对象着手往下处理。怎么做由你决定，但对于我来说，这里采用的做法是最佳的。

4.1.1 从小处着手

首先，回过头去看一下图3-1，想想要实现它该如何做。你首先想到的是什么？也许是符号X和O（如图4-1所示），咱们先来处理它们。



图 4-1

你可以自己制作这些符号，也可以从本书配套网站[1]下载（它们位于示例代码ticTacToev2中）。我使用免费编辑器Inkscape[2]创建了x_blue.png和o_red.png，并将它们保存成了非常大（128

① http://pragprog.com/book/eband4

② http://www.inkscape.org

像素×128像素）的PNG文件。可将PNG文件的背景设置为透明或半透明的，这样可将其放在其他图形的上面。

这些文件存储在目录drawable-xxhdpi中，其中的后缀xxhdpi表示超高（extra extra high）密度，即大约400点每英寸。Android设备还可能装备中、高和极高密度的屏幕，Android会在必要时自动缩放图像。你可以为每种屏幕密度创建一组位图，但就这个游戏而言，那样有点小题大做。有关该资源目录名指定后缀的更详细信息，请参阅8.3.1节。

在Android Studio中，要创建目录，可右击父目录（这里为res），再选择New➤Directory，然后输入目录名。也可以使用Android资源目录向导，但我发现最简单的方式就是最好的。项目窗口的默认模式为Android，在这种模式下，看不到目录drawable-xxhdpi，但它确实存在。如果你尝试将文件粘贴到文件夹drawable中，系统将要求你指定目标目录。

考虑到对XML的介绍还不充分，这里将这些图像封装到一个名为tile.xml的drawable中。前面说过，XML drawable存储在目录drawable中。这个drawable的定义如下。

ticTacToev2/src/main/res/drawable/tile.xml

```
<?xml version="1.0" encoding="utf-8"?>
<level-list xmlns:android="http://schemas.android.com/apk/res/android" >
   <item
      android:drawable="@drawable/x_blue"
      android:maxLevel="0" />
   <item
      android:drawable="@drawable/o_red"
      android:maxLevel="1" />
   <item
      android:drawable="@drawable/tile_empty"
      android:maxLevel="2" />
   <item
      android:drawable="@drawable/tile_available"
      android:maxLevel="3" />
</level-list>
```

每个格子都有4种可能的等级：X、O、空和可下（Available）。等级X和O很容易处理，它们分别显示刚才创建的位图x_blue和o_red。下面是格子为空时显示的drawable的定义。

ticTacToev2/src/main/res/drawable/tile_empty.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
       android:shape="rectangle">
   <stroke
      android:width="@dimen/stroke_width"
      android:color="@color/dark_border_color"/>
   <corners android:radius="@dimen/corner_radius"/>
</shape>
```

它定义了一个带有深色边框的圆角矩形。格子可下时显示的drawable与此类似，但内部使用了绿色（这是通过引用指定的）进行填充。

ticTacToev2/src/main/res/drawable/tile_available.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
       android:shape="rectangle">
   <stroke
      android:width="@dimen/stroke_width"
      android:color="@color/dark_border_color"/>
   <solid android:color="@color/available_color"/>
   <corners android:radius="@dimen/corner_radius"/>
</shape>
```

对于屏幕上的全部81个格子，都将使用tile.xml，但可以通过设置等级，让它们呈现出4种可能外观中的一种。

4.1.2　小棋盘

小棋盘由排列成3×3网格的9个格子组成。我们通过指定行号和列号让每个格子出现在正确的地方。请注意，这些索引从0开始。

ticTacToev2/src/main/res/layout/small_board.xml

```
<GridLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:background="@drawable/tile_background"
   android:elevation="@dimen/elevation_low"
   android:padding="@dimen/small_board_padding"
   tools:context=".GameActivity">

   <ImageButton android:id="@+id/small1" style="@style/TileButton"
                android:layout_column="0" android:layout_row="0"/>
   <ImageButton android:id="@+id/small2" style="@style/TileButton"
                android:layout_column="1" android:layout_row="0"/>
   <ImageButton android:id="@+id/small3" style="@style/TileButton"
                android:layout_column="2" android:layout_row="0"/>
   <ImageButton android:id="@+id/small4" style="@style/TileButton"
                android:layout_column="0" android:layout_row="1"/>
   <ImageButton android:id="@+id/small5" style="@style/TileButton"
                android:layout_column="1" android:layout_row="1"/>
   <ImageButton android:id="@+id/small6" style="@style/TileButton"
                android:layout_column="2" android:layout_row="1"/>
   <ImageButton android:id="@+id/small7" style="@style/TileButton"
                android:layout_column="0" android:layout_row="2"/>
   <ImageButton android:id="@+id/small8" style="@style/TileButton"
                android:layout_column="1" android:layout_row="2"/>
   <ImageButton android:id="@+id/small9" style="@style/TileButton"
                android:layout_column="2" android:layout_row="2"/>
</GridLayout>
```

每个格子都是一个ImageButton。你可能猜到了，ImageButton是图像和按钮的混合体。它是可点击的图像，因为我们需要让玩家通过点击来下棋。

对于每个ImageButton，都指定了行号和列号，还指定了样式TileButton。为定义这种样式，请打开文件styles.xml，并在结束标签</resources>前面添加如下代码。

ticTacToev2/src/main/res/values/styles.xml

```
<style name="TileButton">
   <item name="android:layout_width">@dimen/tile_size</item>
   <item name="android:layout_height">@dimen/tile_size</item>
   <item name="android:layout_margin">@dimen/tile_margin</item>
   <item name="android:background">#00000000</item>
   <item name="android:padding">@dimen/tile_padding</item>
   <item name="android:scaleType">centerCrop</item>
   <item name="android:src">@drawable/tile</item>
</style>
```

这里为何要使用样式呢？必须承认，可以不这样做，但我讨厌不断输入相同的值。这样做并不是想偷懒，而是遵循不自我重复（DRY，Don't Repeat Yourself）原则。有关DRY原则和其他编程技巧的更详细信息，请参阅《程序员修炼之道：从小工到专家》（*The Pragmatic Programmer*）[HT99]。

4.1.3　背景信息

你可能注意到了，小棋盘的定义中包含如下代码行。

```
android:background="@drawable/tile_background"
```

这是做什么的呢？如果你回过头去看图3-1所示的屏幕截图，将发现每个小棋盘都有状态，就像每个格子一样。

tile_background.xml的定义如下。

ticTacToev2/src/main/res/drawable/tile_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<level-list xmlns:android="http://schemas.android.com/apk/res/android">
   <item
      android:drawable="@drawable/tile_blue"
      android:maxLevel="0"/>
   <item
      android:drawable="@drawable/tile_red"
      android:maxLevel="1"/>
   <item
      android:drawable="@drawable/tile_gray"
      android:maxLevel="2"/>
   <item
      android:drawable="@drawable/tile_purple"
      android:maxLevel="3"/>
</level-list>
```

我们用蓝色表示小棋盘被玩家X占据，用红色表示小棋盘被玩家O占据，用灰色表示小棋盘未被任何玩家占据，并使用紫色表示小棋盘被两个玩家占据（即在小棋盘中打成了平手）。这些drawable的定义非常简单。出于完整性考虑，下面将其一一列出。

下面是蓝色背景的定义，用于被玩家X占据的小棋盘。

ticTacToev2/src/main/res/drawable/tile_blue.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
       android:shape="rectangle">
   <stroke
      android:width="@dimen/stroke_width"
      android:color="@color/dark_border_color"/>
   <solid android:color="@color/blue_color"/>
   <corners android:radius="@dimen/corner_radius"/>
</shape>
```

下面是红色背景的定义，用于被玩家O占据的小棋盘。

ticTacToev2/src/main/res/drawable/tile_red.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
       android:shape="rectangle">
   <stroke
      android:width="@dimen/stroke_width"
      android:color="@color/dark_border_color"/>
   <solid android:color="@color/red_color"/>
   <corners android:radius="@dimen/corner_radius"/>
</shape>
```

下面是灰色背景的定义，用于未被任何玩家占据的小棋盘。

ticTacToev2/src/main/res/drawable/tile_gray.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
       android:shape="rectangle">
   <stroke
      android:width="@dimen/stroke_width"
      android:color="@color/dark_border_color"/>
   <solid android:color="@color/gray_color"/>
   <corners android:radius="@dimen/corner_radius"/>
</shape>
```

下面是紫色背景的定义，用于被两个玩家占据的小棋盘。

ticTacToev2/src/main/res/drawable/tile_purple.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
       android:shape="rectangle">
   <stroke
      android:width="@dimen/stroke_width"
```

```
    android:color="@color/dark_border_color"/>
  <solid android:color="@color/purple_color"/>
  <corners android:radius="@dimen/corner_radius"/>
</shape>
```

这些小棋盘背景都是边框为1 dp的圆角矩形，但分别用不同的颜色来填充。

4.1.4 大棋盘

下面来定义大棋盘。大棋盘没什么需要特别指出的，它是由9个小棋盘组成的3×3网格。

ticTacToev2/src/main/res/layout/large_board.xml


```
<GridLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  tools:context=".GameActivity">

  <include android:id="@+id/large1" layout="@layout/small_board"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:layout_margin="@dimen/small_board_margin"
    android:layout_column="0" android:layout_row="0"/>
  <include android:id="@+id/large2" layout="@layout/small_board"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:layout_margin="@dimen/small_board_margin"
    android:layout_column="1" android:layout_row="0"/>
  <include android:id="@+id/large3" layout="@layout/small_board"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:layout_margin="@dimen/small_board_margin"
    android:layout_column="2" android:layout_row="0"/>
  <include android:id="@+id/large4" layout="@layout/small_board"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:layout_margin="@dimen/small_board_margin"
    android:layout_column="0" android:layout_row="1"/>
  <include android:id="@+id/large5" layout="@layout/small_board"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:layout_margin="@dimen/small_board_margin"
    android:layout_column="1" android:layout_row="1"/>
  <include android:id="@+id/large6" layout="@layout/small_board"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:layout_margin="@dimen/small_board_margin"
    android:layout_column="2" android:layout_row="1"/>
  <include android:id="@+id/large7" layout="@layout/small_board"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:layout_margin="@dimen/small_board_margin"
    android:layout_column="0" android:layout_row="2"/>
  <include android:id="@+id/large8" layout="@layout/small_board"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:layout_margin="@dimen/small_board_margin"
    android:layout_column="1" android:layout_row="2"/>
  <include android:id="@+id/large9" layout="@layout/small_board"
```

```
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_margin="@dimen/small_board_margin"
        android:layout_column="2" android:layout_row="2"/>
</GridLayout>
```

其中的标签<include>以前没有见过。它用于创建一个由属性layout指定的布局实例，并设置其他属性，再将视图放到父布局的指定位置。我们原本需要复制并粘贴small_board.xml 9次，但这里采取了偷懒的做法，即遵循DRY原则。

4.1.5　组合在一起

定义完大棋盘后，将它封装到片段中。

ticTacToev2/src/main/res/layout/fragment_game.xml

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    tools:context=".GameActivity">

    <include
        layout="@layout/large_board"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

</RelativeLayout>
```

再将片段和背景图像封装到活动布局中。

ticTacToev2/src/main/res/layout/activity_game.xml

```
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".TicTacToeActivity">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="centerCrop"
        android:src="@drawable/sandy_beach"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="vertical">

        <fragment
```

```
        android:id="@+id/fragment_game"
        class="org.example.tictactoe.GameFragment"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:layout="@layout/fragment_game"/>
    <!--控制片段…… -->
  </LinearLayout>

</FrameLayout>
```

至此，布局就定义好了，余下的工作是编写引用这些布局的Java代码。

4.2 开始游戏

4

暂时切换到文件MainFragment.java。在前一章，我们让About按钮在用户点击时打开一个对话框。这里需要编写这样的代码，即在用户点击New或Continue按钮时启动活动GameActivity。为此，在文件MainFragment.java的方法`onCreateView()`中的`return`语句前面添加如下代码。

ticTacToev2/src/main/java/org/example/tictactoe/MainFragment.java

```
View newButton = rootView.findViewById(R.id.new_button);
View continueButton = rootView.findViewById(R.id.continue_button);
newButton.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {
      Intent intent = new Intent(getActivity(), GameActivity.class);
      getActivity().startActivity(intent);
   }
});
continueButton.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {
      Intent intent = new Intent(getActivity(), GameActivity.class);
      intent.putExtra(GameActivity.KEY_RESTORE, true);
      getActivity().startActivity(intent);
   }
});
```

这些代码首先会在视图层次结构中找到按钮New和Continue，然后设置用户点击按钮时将执行的点击监听器。在每个按钮的监听器中，都新建并启动了一个意图。注意到这里传入了要启动的活动（GameActivity）的名称。因为我们知道这个意图，因此无需查找它。

意图包含活动启动时需要的所有参数。开始新游戏时，不需要任何参数；但继续玩游戏时，需要传递一个标志，指出应接着玩游戏。为此，对意图调用了方法`putExtra()`来设置变量`KEY_RESTORE`的值。GameActivity启动时将读取这个变量的值。

4.2.1 使用快捷键Alt+Enter

注意到，代码`Intent`是红色的，这是因为我们没有导入这个类。在单词Intent上单击，然后

按快捷键Alt+Enter，将自动在文件开头附近添加如下`import`语句，而前述错误也将消失。

ticTacToev2/src/main/java/org/example/tictactoe/MainFragment.java

```
import android.content.Intent;
```

通过使用快捷键Alt+Enter，可自动生成导入类和方法的`import`语句，从而修复大量的错误。

4.2.2　编写GameActivity类

`GameActivity`类的代码大致如下。

ticTacToev2/src/main/java/org/example/tictactoe/GameActivity.java

```
package org.example.tictactoe;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.util.Log;

public class GameActivity extends Activity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_game);
      // 在这里恢复游戏……
   }
}
```

方法`onCreate()`在活动启动时被调用。它根据前面创建的XML文件layout/activity_game.xml创建视图。

说到XML，每次在Android程序中添加新活动时，都必须修改AndroidManifest.xml以引用它。为此，只需在`application`元素末尾前面添加如下代码行即可。

ticTacToev2/src/main/AndroidManifest.xml

```
<activity
   android:name="org.example.tictactoe.GameActivity">
</activity>
```

属性`android:name`的值必须与Java代码中的类名和包名一致。如果包名与清单文件中指定的包名相同，可省略包名，但不能省略类名前面的句点（.）。

1. 继续游戏

我们需要支持继续玩还没有结束的游戏。为此，需要在方法`onCreate()`中添加几行代码，还需给`GameActivity`类声明一些新成员。扩展后的`GameActivity`类的定义如下。

ticTacToev2/src/main/java/org/example/tictactoe/GameActivity.java

```
public static final String KEY_RESTORE = "key_restore";
public static final String PREF_RESTORE = "pref_restore";
private GameFragment mGameFragment;

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_game);
   // 在这里恢复游戏……
   mGameFragment = (GameFragment) getFragmentManager()
         .findFragmentById(R.id.fragment_game);
   boolean restore = getIntent().getBooleanExtra(KEY_RESTORE, false);
   if (restore) {
      String gameData = getPreferences(MODE_PRIVATE)
            .getString(PREF_RESTORE, null);
      if (gameData != null) {
         mGameFragment.putState(gameData);
      }
   }
   Log.d("UT3", "restore = " + restore);
}
```


KEY_RESTORE是传递给新活动的标志的名称，用于将棋盘恢复到以前冻结的状态。首先对Intent实例调用方法getBooleanExtra()，以获取这个标志的值。

如果为true，就使用方法getPreferences()获取指向该活动的Android首选项管理器的句柄，再调用方法getString()获取PREF_RESTORE项的值。首选项非常适合用于永久性存储少量的数据。存储大量数据时，应使用SQLite数据库，详情请参阅第13章。

为了与活动中的游戏片段交流，调用方法getFragmentManager()来获取一个句柄，这个句柄指向跟踪所有片段的对象。接下来，调用方法findFragmentById()来获取指向游戏片段的引用。这并非唯一的方式，也不一定是最佳的方式，但这种方式既实用也管用。从首选项中获取游戏的状态后，便可以调用片段的方法putState()来修改游戏的状态了。

这样，GameActivity启动时，将从首选项中读取游戏的状态。下面来看看游戏状态是如何保存的。

2. 保存游戏

我们希望的行为如下：不管游戏为何停止，即不管是因为用户切换到了主屏幕、启动了另一个应用还是按了返回按钮，我们都希望能够从中断的地方继续。为此，每当活动不再处于运行状态时（参见图2-4所示的生命周期示意图），都需要保存游戏。

从图2-4可知，在方法onPause()中执行这种保存工作非常合适。下面就来在GameActivity类中创建方法onPause()，并添加保存游戏的代码（别忘了，要重新设置代码的格式，使其阅读起来更容易，可使用快捷键Ctrl+Alt+L）。

ticTacToev2/src/main/java/org/example/tictactoe/GameActivity.java

```
@Override
protected void onPause() {
   super.onPause();
   String gameData = mGameFragment.getState();
   getPreferences(MODE_PRIVATE).edit()
         .putString(PREF_RESTORE, gameData)
         .commit();
   Log.d("UT3", "state = " + gameData);
}
```

首先，调用超类的方法onPause()，即Activity.onPause()。重写超类的方法时，都应调用超类的相应方法。对于与初始化相关的所有onXX方法，通常应首先调用超类的相应方法，但对于在终止时调用的方法，应在最后调用超类的相应方法。如果不确定该在什么地方调用超类的相应方法，在开头调用即可。

接下来，调用片段的方法getState()来获取游戏数据。然后，获取一个指向首选项存储区的句柄（getPreferences），为首选项创建一个编辑器（edit），使用键PREF_RESTORE保存游戏数据（putString），并将修改存储到首选项存储区（commit）。Log.d()负责将一条调试消息写入日志。

3. 重新开始游戏

有时候，我们可能想清除所有的记录，一切从头开始。这正是方法restartGame()的目的所在。

ticTacToev2/src/main/java/org/example/tictactoe/GameActivity.java

```
public void restartGame() {
   mGameFragment.restartGame();
}
```

它只是调用稍后将介绍的片段类的同名方法。

4. 宣布获胜方

确定获胜方后，需要采取某种方式进行通告。这项工作是由方法reportWinner()完成的。

ticTacToev2/src/main/java/org/example/tictactoe/GameActivity.java

```
public void reportWinner(final Tile.Owner winner) {
   AlertDialog.Builder builder = new AlertDialog.Builder(this);
   builder.setMessage(getString(R.string.declare_winner, winner));
   builder.setCancelable(false);
   builder.setPositiveButton(R.string.ok_label,
         new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
               finish();
            }
         });
```

```
    final Dialog dialog = builder.create();
    dialog.show();

    // 将棋盘重置为初始状态
    mGameFragment.initGame();
}
```

上述代码使用AlertDialog.Builder类创建了一个消息框，其中包含一行文本和一个OK按钮。用户点击OK按钮时，活动将结束，即关闭游戏屏幕并返回到主菜单。

4.2.3 编写GameFragment类

到目前为止，GameFragment类是最复杂的，因为它需要做的工作很多。下面来定义这个类的轮廓，包括需要的所有import语句。

ticTacToev2/src/main/java/org/example/tictactoe/GameFragment.java

```
package org.example.tictactoe;

import android.app.Fragment;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageButton;

import java.util.HashSet;
import java.util.Set;

public class GameFragment extends Fragment {
   // 在这里定义数据结构……
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      // 设备配置发生变化时保留这个片段
      setRetainInstance(true);
      initGame();
   }
   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                            Bundle savedInstanceState) {
      View rootView =
            inflater.inflate(R.layout.large_board, container, false);
      initViews(rootView);
      updateAllTiles();
      return rootView;
   }
}
```

这里有两个重要的方法。方法onCreate()在创建片段时被调用。在这个方法中，我们调用

了方法setRetainInstance(true)，这样在父活动因设备配置发生变化（如设备旋转）而被销毁时，Android不会销毁该片段。接下来，调用将在后面编写的方法initGame()，以设置所有的数据结构。这个方法将稍后介绍。

虽然这个片段的存活时间比包含它的活动长，但它包含的视图并非如此。方法onCreateView()用于创建（或重新创建）这些视图。它首先调用方法LayoutInflater.inflate()，以读取定义大棋盘的XML并将其转换为视图。接下来，它调用两个方法初始化这些视图并更新格子。这两个方法将在稍后定义。要更详细地了解Android调用片段方法（如onCreate()和onCreateView()）的顺序，请参阅图2-4所示的生命周期示意图。

1. 数据结构

在接着往下介绍前，先在GameFragment类中定义其他方法所需的一些数据结构。这些数据结构应在类定义的开头定义。

ticTacToev2/src/main/java/org/example/tictactoe/GameFragment.java

```
// 在这里定义数据结构……
static private int mLargeIds[] = {R.id.large1, R.id.large2, R.id.large3,
      R.id.large4, R.id.large5, R.id.large6, R.id.large7, R.id.large8,
      R.id.large9,};
static private int mSmallIds[] = {R.id.small1, R.id.small2, R.id.small3,
      R.id.small4, R.id.small5, R.id.small6, R.id.small7, R.id.small8,
      R.id.small9,};

private Tile mEntireBoard = new Tile(this);
private Tile mLargeTiles[] = new Tile[9];
private Tile mSmallTiles[][] = new Tile[9][9];
private Tile.Owner mPlayer = Tile.Owner.X;
private Set<Tile> mAvailable = new HashSet<Tile>();
private int mLastLarge;
private int mLastSmall;
```

mLargeIds和mSmallIds都是常量数组，分别用于将数字映射到小棋盘和格子的资源id。还记得吗，large_board.xml和small_board.xml都定义了9个子视图，这些视图排列成3×3网格。在程序中，将这些子视图编号为0~8：在最上面的那行，子视图的编号为0~2；在中间那行，编号为3~5；在最下面那行，编号为5~8。

mEntireBoard、mLargeTiles和mSmallTiles表示不同层级的格子。在最顶层，只有一个格子，而在最底层，有81个格子，可放入X或O。Tile类可表示任何层级的棋盘格，将在稍后进行定义。

mPlayer存储玩家的id，指出了接下来该谁下。玩家X总是先下。

mAvailable是一个列表，包含给定时点可下的所有格子。这个列表是根据前一步棋和游戏规则计算得到的。

mLastLarge和mLastSmall是最后一步棋的索引。例如，如果它们都为4，则表示最后一步棋下在棋盘的正中央。

2. 初始化游戏

创建片段时，在方法onCreate()中调用了方法initGame()。这个方法将所有的数据结构初始化为起始状态。

ticTacToev2/src/main/java/org/example/tictactoe/GameFragment.java

```
public void initGame() {
   Log.d("UT3", "init game");
   mEntireBoard = new Tile(this);
   // 创建所有的格子
   for (int large = 0; large < 9; large++) {
      mLargeTiles[large] = new Tile(this);
      for (int small = 0; small < 9; small++) {
         mSmallTiles[large][small] = new Tile(this);
      }
      mLargeTiles[large].setSubTiles(mSmallTiles[large]);
   }
   mEntireBoard.setSubTiles(mLargeTiles);

   // 设置先下棋子的玩家可下的格子
   mLastSmall = -1;
   mLastLarge = -1;
   setAvailableFromLastMove(mLastSmall);
}
```

这个方法用于创建一个表示整个棋盘的Tile实例、9个表示小棋盘的Tile实例和81个表示格子的Tile实例。接下来，它会假设出上一步棋，并计算哪些格子可下（应该是全部格子）。

3. 初始化视图

初始化视图的代码如下。

ticTacToev2/src/main/java/org/example/tictactoe/GameFragment.java

```
private void initViews(View rootView) {
   mEntireBoard.setView(rootView);
   for (int large = 0; large < 9; large++) {
      View outer = rootView.findViewById(mLargeIds[large]);
      mLargeTiles[large].setView(outer);

      for (int small = 0; small < 9; small++) {
         ImageButton inner = (ImageButton) outer.findViewById
               (mSmallIds[small]);
         final int fLarge = large;
         final int fSmall = small;
         final Tile smallTile = mSmallTiles[large][small];
         smallTile.setView(inner);
         inner.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View view) {
               if (isAvailable(smallTile)) {
                  makeMove(fLarge, fSmall);
                  switchTurns();
               }
            }
         });
      }
   }
}
```

依次处理整个棋盘、小棋盘和格子。设置完整个棋盘、小棋盘和格子的视图后，来设置用户点击格子时将调用的点击处理程序：如果当前格子可下，就将棋下在该格子中，并让另一个玩家接着下。

4. 将棋下到格子中

为了将棋下到格子中，我们调用方法makeMove()，并传入相应的小棋盘索引和格子索引。

ticTacToev2/src/main/java/org/example/tictactoe/GameFragment.java

```
private void makeMove(int large, int small) {
   mLastLarge = large;
   mLastSmall = small;
   Tile smallTile = mSmallTiles[large][small];
   Tile largeTile = mLargeTiles[large];
   smallTile.setOwner(mPlayer);
   setAvailableFromLastMove(small);
   Tile.Owner oldWinner = largeTile.getOwner();
   Tile.Owner winner = largeTile.findWinner();
   if (winner != oldWinner) {
      largeTile.setOwner(winner);
   }
   winner = mEntireBoard.findWinner();
   mEntireBoard.setOwner(winner);
   updateAllTiles();
   if (winner != Tile.Owner.NEITHER) {
      ((GameActivity)getActivity()).reportWinner(winner);
   }
}
```

这里做的主要工作是，让当前玩家占据格子。接下来，需要确定是否有玩家赢得了当前格子所属的小棋盘。如果有，就设置该小棋盘的占据者。最后，检查整个棋盘，看是否有玩家占据了整个棋盘。如果已经有玩家占据了整个棋盘，就意味着该玩家获得了胜利，需要调用宣告获胜者的方法。

5. 让另一个玩家接着下

这实现起来很容易，只需切换变量mPlayer的值即可。

ticTacToev2/src/main/java/org/example/tictactoe/GameFragment.java

```
private void switchTurns() {
   mPlayer = mPlayer == Tile.Owner.X ? Tile.Owner.O : Tile
         .Owner.X;
}
```

如果当前玩家为X，就将接下来的玩家设置为O，否则将接下来的玩家设置为X。

6. 重新开始游戏

用户点击Restart按钮时，监听器将调用`GameActivity`类的方法`restartGame()`，而这个方法将转而调用游戏片段的方法`restartGame()`。

ticTacToev2/src/main/java/org/example/tictactoe/GameFragment.java

```
public void restartGame() {
   initGame();
   initViews(getView());
   updateAllTiles();
}
```

这个方法首先初始化游戏状态，就像游戏刚启动时那样。接下来，它会初始化视图，并更新所有的格子以便正确地绘制它们。

7. 计算可下棋的格子

根据终极版井字游戏的规则，一个玩家下棋后，对手接下来只能在这步棋所处的格子对应的小棋盘内下棋。例如，如果玩家1下的棋位于某个小棋盘的左上角，那么玩家2接下来只能在左上角的小棋盘内下棋。如果没有可下棋的格子（即指定的小棋盘已满），则对手可在任何地方下棋。

使用多个方法来根据前一步棋计算接下来可在哪些格子中下棋。

ticTacToev2/src/main/java/org/example/tictactoe/GameFragment.java

```
private void clearAvailable() {
   mAvailable.clear();
}

private void addAvailable(Tile tile) {
   mAvailable.add(tile);
}

public boolean isAvailable(Tile tile) {
   return mAvailable.contains(tile);
}

private void setAvailableFromLastMove(int small) {
   clearAvailable();
   // 让目标小棋盘中所有空格子都可下棋
   if (small != -1) {
      for (int dest = 0; dest < 9; dest++) {
         Tile tile = mSmallTiles[small][dest];
```

```
        if (tile.getOwner() == Tile.Owner.NEITHER)
           addAvailable(tile);
     }
  }
  // 如果目标小棋盘没有空格子，则令整个棋盘的所有空格子都可下棋
  if (mAvailable.isEmpty()) {
     setAllAvailable();
  }
}

private void setAllAvailable() {
   for (int large = 0; large < 9; large++) {
      for (int small = 0; small < 9; small++) {
         Tile tile = mSmallTiles[large][small];
         if (tile.getOwner() == Tile.Owner.NEITHER)
            addAvailable(tile);
      }
   }
}
```

clearAvailable()用于清空可下棋格子列表，以便能够在其中添加格子；addAvailable()用于将一个格子加入可下棋格子列表中；isAvailable()用于判断指定的格子是否可下棋；setAvailableFromLastMove()用于清空可下棋格子列表，并将目标小棋盘中所有的空格子都标记为可下棋的。最后，如果目标小棋盘中没有空格子，就调用方法setAllAvailable()，将整个棋盘中的所有空格子都标记为可下棋的。

8. 处理状态

为了将当前游戏状态保存下来，以便以后能够继续游戏，需要将游戏的当前状态转换为序列化形式（一个字符串），以便将其存储到首选项存储区中。下面是创建这个字符串的代码。

ticTacToev2/src/main/java/org/example/tictactoe/GameFragment.java

```
/** 创建包含游戏状态的字符串 */
public String getState() {
   StringBuilder builder = new StringBuilder();
   builder.append(mLastLarge);
   builder.append(',');
   builder.append(mLastSmall);
   builder.append(',');
   for (int large = 0; large < 9; large++) {
      for (int small = 0; small < 9; small++) {
         builder.append(mSmallTiles[large][small].getOwner().name());
         builder.append(',');
      }
   }
   return builder.toString();
}
```

最重要的是，明确每个格子都由哪个玩家占据，但还必须跟踪上一步棋，因为接下来玩家可在哪些地方下棋就取决于上一步棋。

创建字符串并将其保存到首选项存储区后，便可提取该字符串并将其反序列化为新的游戏状态。这是由方法putState()完成的。

ticTacToev2/src/main/java/org/example/tictactoe/GameFragment.java

```
/** 根据给定的字符串恢复游戏状态 */
public void putState(String gameData) {
   String[] fields = gameData.split(",");
   int index = 0;
   mLastLarge = Integer.parseInt(fields[index++]);
   mLastSmall = Integer.parseInt(fields[index++]);
   for (int large = 0; large < 9; large++) {
      for (int small = 0; small < 9; small++) {
         Tile.Owner owner = Tile.Owner.valueOf(fields[index++]);
         mSmallTiles[large][small].setOwner(owner);
      }
   }
   setAvailableFromLastMove(mLastSmall);
   updateAllTiles();
}
```

这个方法基本上与getState()相反。这里要获取上一步棋和每个格子的占据者，并将它们存储到成员变量中。最后，重新计算可下棋格子列表，并更新所有格子的drawable状态。

在序列化游戏状态时，丢失了一些信息。你知道都是哪些信息吗？

下面是更新drawable状态的代码。

ticTacToev2/src/main/java/org/example/tictactoe/GameFragment.java

```
private void updateAllTiles() {
   mEntireBoard.updateDrawableState();
   for (int large = 0; large < 9; large++) {
      mLargeTiles[large].updateDrawableState();
      for (int small = 0; small < 9; small++) {
         mSmallTiles[large][small].updateDrawableState();
      }
   }
}
```

上述代码用于更新表示整个棋盘、9个小棋盘和每个格子（包含X或O）的Tile实例的状态。

4.2.4　定义Tile类

Tile类表示任何层次的棋盘格子，它可以是可包含X或O的最小格子、包含9个格子的小棋盘或包含9个小棋盘的大棋盘。

下面是Tile类的轮廓。

ticTacToev2/src/main/java/org/example/tictactoe/Tile.java

```
package org.example.tictactoe;
```

```
import android.graphics.drawable.Drawable;
import android.view.View;
import android.widget.ImageButton;

public class Tile {

   public enum Owner {
      X, O /* 字母O */, NEITHER, BOTH
   }

   // 这些级别是在drawable中定义的
   private static final int LEVEL_X = 0;
   private static final int LEVEL_O = 1; // letter O
   private static final int LEVEL_BLANK = 2;
   private static final int LEVEL_AVAILABLE = 3;
   private static final int LEVEL_TIE = 3;

   private final GameFragment mGame;
   private Owner mOwner = Owner.NEITHER;
   private View mView;
   private Tile mSubTiles[];

   public Tile(GameFragment game) {
      this.mGame = game;
   }

   public View getView() {
      return mView;
   }

   public void setView(View view) {
      this.mView = view;
   }

   public Owner getOwner() {
      return mOwner;
   }

   public void setOwner(Owner owner) {
      this.mOwner = owner;
   }

   public Tile[] getSubTiles() {
      return mSubTiles;
   }

   public void setSubTiles(Tile[] subTiles) {
      this.mSubTiles = subTiles;
   }
}
```

Tile对象包含占据者和视图（图形表示），还可能包含一系列子格子。Tile类包含构造函数

`Tile`，还包括占据者、视图和子格子的获取函数和设置函数。另外，它还包含管理drawable状态的代码以及判断谁是占据者的代码。下面来看一下管理drawable状态的代码。这些代码应放在`Tile`类的结束大括号之前。

ticTacToev2/src/main/java/org/example/tictactoe/Tile.java

```
public void updateDrawableState() {
   if (mView == null) return;
   int level = getLevel();
   if (mView.getBackground() != null) {
      mView.getBackground().setLevel(level);
   }
   if (mView instanceof ImageButton) {
      Drawable drawable = ((ImageButton) mView).getDrawable();
      drawable.setLevel(level);
   }
}

private int getLevel() {
   int level = LEVEL_BLANK;
   switch (mOwner) {
      case X:
         level = LEVEL_X;
         break;
      case O: // 字母O
         level = LEVEL_O;
         break;
      case BOTH:
         level = LEVEL_TIE;
         break;
      case NEITHER:
         level = mGame.isAvailable(this) ? LEVEL_AVAILABLE : LEVEL_BLANK;
         break;
   }
   return level;
}
```

在4.1.1节中讲过，最小的格子有4种等级，具体是哪种等级取决于其占据者：X、O、两者和无。方法`updateDrawableState()`调用`getLevel()`来确定等级，然后根据视图的类型，对视图背景或drawable调用方法`setLevel()`。

接下来，需要编写确定占据者的代码。

ticTacToev2/src/main/java/org/example/tictactoe/Tile.java

```
public Owner findWinner() {
   // 如果已确定占据者，就返回它
   if (getOwner() != Owner.NEITHER)
      return getOwner();

   int totalX[] = new int[4];
   int totalO[] = new int[4];
```

```
        countCaptures(totalX, totalO);
        if (totalX[3] > 0) return Owner.X;
        if (totalO[3] > 0) return Owner.O;

        // 检查是否打成了平局
        int total = 0;
        for (int row = 0; row < 3; row++) {
            for (int col = 0; col < 3; col++) {
                Owner owner = mSubTiles[3 * row + col].getOwner();
                if (owner != Owner.NEITHER) total++;
            }
            if (total == 9) return Owner.BOTH;
        }

        // 未被任何玩家占据
        return Owner.NEITHER;
    }
```

如果已经有占据者，就返回该占据者；否则，计算两位玩家占据的格子数。如果有一个玩家占据了3个排成一条线的格子，那么该玩家就是占据者；否则检查小棋盘的所有格子是否都不为空。如果是这样，说明打成了平局，因此返回`BOTH`；否则，说明未被任何玩家占据，因此返回`NEITHER`。

方法`countCaptures()`的定义如下。

ticTacToev2/src/main/java/org/example/tictactoe/Tile.java

```
private void countCaptures(int totalX[], int totalO[]) {
    int capturedX, capturedO;
    // 检查是否有同一个玩家的3个棋子排成了一行
    for (int row = 0; row < 3; row++) {
        capturedX = capturedO = 0;
        for (int col = 0; col < 3; col++) {
            Owner owner = mSubTiles[3 * row + col].getOwner();
            if (owner == Owner.X || owner == Owner.BOTH) capturedX++;
            if (owner == Owner.O || owner == Owner.BOTH) capturedO++;
        }
        totalX[capturedX]++;
        totalO[capturedO]++;
    }

    // 检查是否有同一个玩家的3个棋子排成了一列
    for (int col = 0; col < 3; col++) {
        capturedX = capturedO = 0;
        for (int row = 0; row < 3; row++) {
            Owner owner = mSubTiles[3 * row + col].getOwner();
            if (owner == Owner.X || owner == Owner.BOTH) capturedX++;
            if (owner == Owner.O || owner == Owner.BOTH) capturedO++;
        }
        totalX[capturedX]++;
        totalO[capturedO]++;
    }
```

```
    // 检查是否有同一个玩家的3个棋子排成对角线
    capturedX = capturedO = 0;
    for (int diag = 0; diag < 3; diag++) {
       Owner owner = mSubTiles[3 * diag + diag].getOwner();
       if (owner == Owner.X || owner == Owner.BOTH) capturedX++;
       if (owner == Owner.O || owner == Owner.BOTH) capturedO++;
    }
    totalX[capturedX]++;
    totalO[capturedO]++;
    capturedX = capturedO = 0;
    for (int diag = 0; diag < 3; diag++) {
       Owner owner = mSubTiles[3 * diag + (2 - diag)].getOwner();
       if (owner == Owner.X || owner == Owner.BOTH) capturedX++;
       if (owner == Owner.O || owner == Owner.BOTH) capturedO++;
    }
    totalX[capturedX]++;
    totalO[capturedO]++;
}
```


代码看起来有很多，但其实非常简单。首先，计算各行的X和O数量，然后计算各列的X和O数量，最后计算两条对角线上的X和O数量。结果是通过两个数组返回的：表示玩家X的数组`totalX`和表示玩家O的数组`totalO`。

4.3　控制游戏

玩这个游戏时，用户可能想重新开始游戏或返回到主菜单。当然，用户可以按返回按钮来返回主菜单，但鉴于有些人不知道返回按钮在哪里，我们将提供一种返回主菜单的途径。

先来看包含按钮Restart和Main Menu的片段。

ticTacToev2/src/main/res/layout/fragment_control.xml

```
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:orientation="horizontal"
   android:padding="@dimen/control_padding"
   tools:context=".GameActivity">

   <Button
      android:id="@+id/button_restart"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:elevation="@dimen/elevation_low"
      android:drawableTop="@drawable/restart"
      android:text="@string/restart_label"/>

   <Button
      android:id="@+id/button_main"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:elevation="@dimen/elevation_low"
        android:drawableTop="@drawable/home"
        android:text="@string/main_menu_label"/>

</LinearLayout>
```

两幅图像可放在任何drawable目录中。我使用绘图程序绘制了这两幅图像，将它们命名为home.png和restart.png，并将其与GameActivity背景图像（sandy_beach.jpg）一起放在了目录res/drawable-xxhdpi中。在本书配套网站的示例代码压缩文件中，可找到所有这些图像。

接下来，需要在GameActivity中包含一个新片段。

ticTacToev2/src/main/res/layout/activity_game.xml

```
<!-- 控制片段…… -->
<fragment
   android:id="@+id/fragment_game_controls"
   class="org.example.tictactoe.ControlFragment"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   tools:layout="@layout/fragment_control"/>
```

最后，需要编写一些处理该布局的Java代码。

ticTacToev2/src/main/java/org/example/tictactoe/ControlFragment.java

```
package org.example.tictactoe;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class ControlFragment extends Fragment {

   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                            Bundle savedInstanceState) {
      View rootView =
            inflater.inflate(R.layout.fragment_control, container, false);
      View main = rootView.findViewById(R.id.button_main);
      View restart = rootView.findViewById(R.id.button_restart);

      main.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            getActivity().finish();
         }
      });
      restart.setOnClickListener(new View.OnClickListener() {
         @Override
```

```
            public void onClick(View view) {
               ((GameActivity) getActivity()).restartGame();
            }
         });
         return rootView;
      }
   }
```

通过Main Menu按钮结束当前活动，与按返回按钮的效果相同。

Restart按钮假定控制片段嵌入在GameActivity中，因此会将当前活动转换为GameActivity，并调用前面定义的重新开始游戏的方法。

4.4 支持横向模式

本章使用了一些尺寸和颜色，下面来定义这些资源，以消除所有相关的错误。首先来定义尺寸。

ticTacToev2/src/main/res/values/dimens.xml

```
<dimen name="activity_horizontal_margin">8dp</dimen>
<dimen name="activity_vertical_margin">8dp</dimen>
<dimen name="tile_size">30dp</dimen>
<dimen name="tile_margin">0dp</dimen>
<dimen name="tile_padding">3dp</dimen>
<dimen name="control_padding">20dp</dimen>
<dimen name="small_board_padding">2dp</dimen>
<dimen name="small_board_margin">2dp</dimen>
<dimen name="elevation_low">4dp</dimen>
```

你可能会问，如何确定边距、格子大小和其他尺寸呢？我本应该使用绘图纸进行规划的，但实际情况是，我只是在不断地调整和优化，直到获得满意效果为止。不要把什么事情都搞得那么复杂。

接下来定义颜色。

ticTacToev2/src/main/res/values/colors.xml

```
<color name="dark_border_color">#4f4f4f</color>
<color name="available_color">#7fbf7f</color>
<color name="blue_color">#7f7fff</color>
<color name="gray_color">#bfbfbf</color>
<color name="purple_color">#7f007f</color>
<color name="red_color">#ff7f7f</color>
```

在选择各种颜色时，只要它们与背景图像以及X和O的颜色相配即可。

最后，别忘了字符串。

ticTacToev2/src/main/res/values/strings.xml

```
<string name="restart_label">Restart</string>
<string name="main_menu_label">Main Menu</string>
<string name="declare_winner">%1$s is the winner</string>
```

既然在定义资源，就顺便定义GameActivity和控制片段的横向版本吧，它们将在用户改变设备朝向时使用。横向版本与纵向版本类似，但前者经过了重新排列，以使其更适合宽屏幕。

ticTacToev2/src/main/res/layout-land/activity_game.xml

```
<FrameLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".TicTacToeActivity">

   <ImageView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:scaleType="centerCrop"
      android:src="@drawable/sandy_beach"/>

   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:gravity="center"
      android:baselineAligned="false"
      android:orientation="horizontal">

      <fragment
         android:id="@+id/fragment_game"
         class="org.example.tictactoe.GameFragment"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         tools:layout="@layout/fragment_game"/>

      <fragment
         android:id="@+id/fragment_game_controls"
         class="org.example.tictactoe.ControlFragment"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         tools:layout="@layout/fragment_control"/>
   </LinearLayout>

</FrameLayout>
```

ticTacToev2/src/main/res/layout-land/fragment_control.xml

```
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
```

```
    android:orientation="vertical"
    android:padding="@dimen/control_padding"
    tools:context=".GameActivity">

    <Button
        android:id="@+id/button_restart"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:drawableTop="@drawable/restart"
        android:elevation="@dimen/elevation_low"
        android:text="@string/restart_label"/>

    <Button
        android:id="@+id/button_main"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:drawableTop="@drawable/home"
        android:elevation="@dimen/elevation_low"
        android:text="@string/main_menu_label"/>

</LinearLayout>
```

请注意，这些版本存储在目录res/layout-land中，其中的后缀-land表示横向模式。

现在，应该能够运行这个游戏了，同时还能实现轮流下棋，直到一方获胜。请找个朋友一起试一试。你将发现想要获胜并没有你想象的那么容易。

如果遇到了麻烦，那就当作一次学习经验吧。首先消除所有的编译器错误，然后参考3.6节来消除运行阶段错误，如异常和空屏幕。

如果实在不行，可从本书配套网站下载代码，并将其与你的代码进行比较。为此，先下载并解压缩ZIP文件，再在项目窗口中右击文件夹app，并选择Compare Directory with...，然后切换到示例ticTacToev2的目录src，并单击OK按钮。

4.5 快速阅读指南

在本章中，你学习了如何从零开始创建用户界面：从最小的单元（格子）着手，依次向上创建小棋盘和大棋盘。在布局中还包含了其他布局，需要通过定义样式来避免自我重复。你还学习了一种在片段和活动之间通信的方式。

下一章将给这款游戏添加AI，让它成为非常厉害的对手。如果你想跳过这部分，可以从配套网站下载相关的代码，然后直接阅读介绍多媒体的第6章以及介绍动画的第7章。

第 5 章 机器幽灵

到目前为止，我们创建的游戏的玩法是，玩家X和O通过触摸屏幕交替地下棋。在本章中，我们将把它变成单玩家游戏，由玩家和计算机对玩。

为此，需要让这个应用知道如何评估当前棋局，并从所有的下法中选择最佳的下法。因此，我们需要暂时离开正题，来简单地认识一下AI（人工智能）。

5.1 AI 简介

在你所想编写的任何游戏中，AI都扮演着至关重要的角色，在有些非游戏程序中亦如此。例如，AI被用于创建智能应用，它们能够根据以往的经验预测用户的行为。由于用户的能力常常被投射到程序中，因此实现这样的人工智能并不是很难。

5.1.1 AI的工作原理

过去几年，计算机科学家开发了很多教计算机如何玩游戏的方法，其中一种最简单的方法是minimax算法。这个算法因其根据接下来轮到谁下交替地最小化或最大化得分而得名。

玩游戏的过程被分成一系列的步（半轮）。玩家下棋时，是一步；计算机下棋时，是另一步。

轮到计算机下棋时，将调用minimax算法。它会对每种走法进行评估（这是通过给相应的棋局分配一个数字实现的）。这是第一阶段。对于有利于玩家的棋局，将随意分配一个正数；而对于有利于计算机的棋局，将分配一个负数。因此，最佳的走法就是数字最小的走法。

接下来，对于每种走法，计算机都将考虑玩家的所有应对走法，并对这些走法形成的棋局进行评估。这是第二阶段。玩家自然想选择最佳的走法，因此在第二个阶段，最佳的走法是数字最大的走法。

这个算法将不断地运行。可以想见，它不能看很多步，因为每多看一步，需要评估的棋局数量就会有所增加，增加的倍数为这一步的可能走法数。例如，如果要看3步，而每步都有5种可能的走法，就需要评估5 × 5 × 5 = 125个棋局。如果试图看14步（7轮），需要评估的棋局数将多达

数10亿。通常来讲，可供计算机思考的时间是有限的，因此只能考虑一定的步数，并从中选择最佳的结果。

为减少计算量，可使用剪枝算法（alpha-beta pruning）①和Negamax②等算法，但这些算法不在本书的讨论范围内。实际上，在这个示例中，我们只评估一步。

5.1.2　形势判断

人类通过观看棋盘就能做出形势判断，但计算机没有这样的能力，它们只能理解数字。如何将棋局转换为数字呢？这是评估函数的工作。给游戏添加AI时，创建良好的评估函数至关重要。请看下面的评估函数。

ticTacToev3/src/main/java/org/example/tictactoe/Tile.java

```
public int evaluate() {
   switch (getOwner()) {
      case X:
         return 100;
      case O:
         return -100;
      case NEITHER:
         int total = 0;
         if (getSubTiles() != null) {
            for (int tile = 0; tile < 9; tile++) {
               total += getSubTiles()[tile].evaluate();
            }
            int totalX[] = new int[4];
            int totalO[] = new int[4];
            countCaptures(totalX, totalO);
            total = total * 100 + totalX[1] + 2 * totalX[2] + 8 *
                  totalX[3] - totalO[1] - 2 * totalO[2] - 8 * totalO[3];
         }
         return total;
   }
   return 0;
}
```

终极版井字游戏的棋盘是一个层次结构。最顶层是整个棋盘，包含9个小棋盘，而每个小棋盘又包含9个格子。格子没有子元素，它包含X或O。

如果整个棋盘、小棋盘或格子被X或O玩家占据，上述评估函数将返回一个很大的数字。有趣的是未被任何玩家占据的情况。在这种情况下，将查看下一层。

首先评估每个子元素，并将每个子元素的评估结果相加。接下来考虑子元素的占据情况，并据此计算一个公式的值：对于X占据的子元素，每出现3个排成一条线的情况时都加8；每出现两

① http://en.wikipedia.org/wiki/Alpha-beta_pruning

② http://en.wikipedia.org/wiki/Negamax

个子元素排除一条线时都加2；每个单独的子元素都加1；对于O占据的子元素，采用相同的算法，但不是加上而是减去相应的数字。

上述评估函数使用的公式和数字完全是随意选择的。作为练习，你能设计出更好的评估函数吗？这应该不会太难。

5.2　模拟思考过程

在实际游戏中，需要看很多步，直到游戏结束（输、赢或平）或定时器到期。在这个示例中，我们将使用定时器来模拟一秒钟的考虑时间。

5.2.1　使用Handler和postDelayed

首先，需要在GameFragment.java中声明并初始化一个Android Handler实例。

ticTacToev3/src/main/java/org/example/tictactoe/GameFragment.java

```
import android.os.Handler;
public class GameFragment extends Fragment {
   // 在这里定义数据结构……
   private Handler mHandler = new Handler();
   // ……
}
```

Handler类能够将事情推迟到以后再做。我们将调用它的方法postDelayed()，并传入定时器到期后要运行的代码以及要等待的毫秒数。最重要的是，这些代码将在用户界面线程（主线程）中执行，因此可以操作屏幕上的视图。

为此，首先修改点击监听器，在其中调用新方法think()。

ticTacToev3/src/main/java/org/example/tictactoe/GameFragment.java

```
private void initViews(View rootView) {
         // ……
         inner.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
               if (isAvailable(smallTile)) {
                  makeMove(fLarge, fSmall);
                  think();
               }
            }
         });
         // ……
}
```

方法think()的定义如下。

ticTacToev3/src/main/java/org/example/tictactoe/GameFragment.java

```
private void think() {
   ((GameActivity)getActivity()).startThinking();
   mHandler.postDelayed(new Runnable() {
      @Override
      public void run() {
         if (getActivity() == null) return;
         if (mEntireBoard.getOwner() == Tile.Owner.NEITHER) {
            int move[] = new int[2];
            pickMove(move);
            if (move[0] != -1 && move[1] != -1) {
               switchTurns();
               makeMove(move[0], move[1]);
               switchTurns();
            }
         }
         ((GameActivity) getActivity()).stopThinking();
      }
   }, 1000);
}
```


用户轻按棋盘格时，开启思考指示器，执行用户要求下的棋，然后启动定时器。1000毫秒（1秒）后，执行方法run()。

如果等待期间活动未关闭且游戏未结束，就来执行下一步棋，然后关闭思考指示器。仅此而已！准确地说，是差不多仅此而已。

5.2.2 在思考期间阻断输入

刚编写出来时，程序最初运行得很好，但如果轻按棋盘格的速度太快，程序就崩溃了。这是因为，我在思考期间轻按了棋盘格和按钮，导致代码没有按正确的顺序执行。如何避免这种情况的发生呢?

一种办法是，在每个接受输入的地方都进行检查。如果当前为思考时间，就禁止轻按操作进入队列。还有一种更简单的方法，那就是在思考期间阻断所有的输入。可以巧妙地利用思考指示器来达到这个目的。并不是要将指示器设计得很小，只占据一角，而是要将其设计为覆盖整个屏幕的大视图，从而拦截所有的轻按操作。可以在该视图的一角放置进度指示器，其他部分均设置为透明的。图5-1是我们希望游戏陷入思考时用户看到的界面。

该思考视图的布局如下。

ticTacToev3/src/main/res/layout/thinking.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/thinking"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:clickable="true"
    android:visibility="gone"
    tools:showIn="@layout/game_activity">

    <ProgressBar
        android:background="@drawable/thinking_background"
        android:elevation="@dimen/elevation_high"
        android:layout_marginTop="@dimen/activity_vertical_margin"
        android:layout_marginLeft="@dimen/activity_vertical_margin"
        android:layout_width="@dimen/thinking_progress_size"
        android:layout_height="@dimen/thinking_progress_size"
        android:indeterminate="true"/>
</FrameLayout>
```



图　5-1

在GameActivity的布局中，在`FrameLayout`结束标签前面添加下面一行。

ticTacToev3/src/main/res/layout/activity_game.xml

```
<include layout="@layout/thinking"/>
```

进度条的背景是一个椭圆。

ticTacToev3/src/main/res/drawable/thinking_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">

    <solid android:color="@color/thinking_background_color"/>
</shape>
```

现在，可以在GameActivity中使用这个视图了。

ticTacToev3/src/main/java/org/example/tictactoe/GameActivity.java

```
import android.view.View;

public class GameActivity extends Activity {
   // ……
   public void startThinking() {
      View thinkView = findViewById(R.id.thinking);
      thinkView.setVisibility(View.VISIBLE);
   }

   public void stopThinking() {
      View thinkView = findViewById(R.id.thinking);
      thinkView.setVisibility(View.GONE);
   }
}
```

这里有两个方法：`startThinking()`和`stopThinking()`，它们都可以获取我们在activity_game.xml中添加的思考视图，然后将它设置为可见或消失（Gone）。消失类似于不可见，但不接受任何事件，也不加入到布局中。

5.3 下棋

要选择接下来的走法，只需评估每种可能走法导致的棋局，并选择得分最低的走法即可（别忘了，在计算机看来，低就是好）。

5.3.1 选择正确的走法

在GameFragment.java中，添加下述选择走法的代码。

ticTacToev3/src/main/java/org/example/tictactoe/GameFragment.java

```
private void pickMove(int move[]) {
   Tile.Owner opponent = mPlayer == Tile.Owner.X ? Tile.Owner.O : Tile
         .Owner.X;
   int bestLarge = -1;
   int bestSmall = -1;
   int bestValue = Integer.MAX_VALUE;
   for (int large = 0; large < 9; large++) {
      for (int small = 0; small < 9; small++) {
         Tile smallTile = mSmallTiles[large][small];
         if (isAvailable(smallTile)) {
            // 尝试下棋并评估得到的棋局的得分
            Tile newBoard = mEntireBoard.deepCopy();
            newBoard.getSubTiles()[large].getSubTiles()[small]
                  .setOwner(opponent);
            int value = newBoard.evaluate();
            Log.d("UT3",
                  "Moving to " + large + ", " + small + " gives value " +
                        "" + value
```

```
            );
            if (value < bestValue) {
               bestLarge = large;
               bestSmall = small;
               bestValue = value;
            }
         }
      }
   }
   move[0] = bestLarge;
   move[1] = bestSmall;
   Log.d("UT3", "Best move is " + bestLarge + ", " + bestSmall);
}
```

遍历每个棋盘格（总共81个），并使用方法`isAvailable()`判断是否可在该棋盘格中下棋。如果可以，就复制整个棋盘，调用方法`setOwner()`在该棋盘格中下棋，然后再评估棋局。在遍历过程中，需要记录评估函数返回的最佳值及对应的走法。循环结束后，通过传入的数组返回走法（这是一个返回多个值的Java成例）。

下面是复制棋盘的代码，它包含在Tile.java中。

ticTacToev3/src/main/java/org/example/tictactoe/Tile.java

```
public Tile deepCopy() {
   Tile tile = new Tile(mGame);
   tile.setOwner(getOwner());
   if (getSubTiles() != null) {
      Tile newTiles[] = new Tile[9];
      Tile oldTiles[] = getSubTiles();
      for (int child = 0; child < 9; child++) {
         newTiles[child] = oldTiles[child].deepCopy();
      }
      tile.setSubTiles(newTiles);
   }
   return tile;
}
```

首先创建一个新的`Tile`实例，并复制占据者。接下来，检查它是否有子元素。如果有，就创建一个新的`Tile`引用数组，用来存储子元素副本。然后，对每个子元素递归调用`deepCopy()`。最后，将子元素引用设置为这个新数组。如果最初的`Tile`实例没有子元素（换句话说，它是包含X或O的棋盘格），那就什么都不做。

5.3.2　颜色和尺寸

如果回过头去看看前面的示例代码，将发现其中包含颜色和尺寸引用。下面是我给它们指定的值，你也可以使用你认为合适的任何值。

思考指示器的背景为浅灰色。

ticTacToev3/src/main/res/values/colors.xml

```
<color name="thinking_background_color">#cfdfdfdf</color>
```

指示器的尺寸是在res/values/dimens.xml中设置的。

ticTacToev3/src/main/res/values/dimens.xml

```
<dimen name="thinking_progress_size">50dp</dimen>
```

指示器本身大小适中，它位于屏幕一角，既没有大到分散用户注意力的程度，又不会太小，在大多数设备上都能看得到。

5.4 快速阅读指南

在本章中，我们给游戏添加了下棋的功能。这里稍微谈及了博弈论，并制作了一个在游戏思考期间显示的指示器。接下来的两章将给这个井字游戏添加动画和声音，如果你对此不感兴趣，可直接跳到第8章，学习如何使应用能够在任何设备上运行。

第 6 章 添加声音

还记得苹果的电视广告吗？一个剪影跟随着iPod播放的音乐节奏狂舞。你一定希望自己的作品也像这样激动人心。当然，超过18岁的常人无法这样跳舞，除非有人将一条蜥蜴放进他的衣服里——不好意思，我有点离题了。相比于仅使用文本和图形，添加音乐和音效后，程序更能让人沉醉其中，不能自拔。

本章介绍如何在Android应用中添加多媒体。这也许不能让用户乐不可支，但只要处理妥当，至少能够让他们面露笑容。

6.1 音乐之声

那是一个狂风大作的漆黑夜晚……发令枪响起，他们起跑了……离比赛结束还有一秒钟，斯泰特投进了一个三分球，人群沸腾了……

音乐会弥漫在整个环境中，影响人的情绪。声音是向用户传递信息的另一种方式。你可以在屏幕上显示图形来向用户传递信息，并将音频作为强化信息传递的辅助手段。

Android支持音乐播放，这是通过android.media包[①]中的`MediaPlayer`类实现的。下面来给井字游戏的开始场景添加背景乐。

为此，首先得有音乐。你可以使用自己喜欢的任何音乐。freesound.org[②]是一个极佳的音频搜索网站，你可以搜索特定的音频类型和流派，试听、下载喜欢的音频，并在自己的应用中使用它们。如果你要发布应用，务必遵循音频版权许可，将其用于商业目时尤其要注意。

我选择了用户a_guy_1制作的音频epicbuilduploop，将其用作开场乐。这个音频最初的格式为MP3。Android能够播放MP3，但我发现，需要循环播放或剪辑很短时，使用OGG格式的效果最佳，因为起点和终点更精确。有鉴于此，我使用了程序Audacity[③]对这个示例使用的所有声音进行

---

① http://d.android.com/guide/topics/media

② http://www.freesound.org

③ http://audacity.sourceforge.net

了编辑和转换。在本书配套网站[①]的示例代码ticTacToev4中，可以找到转换后的音乐和音效。

挑选并转换音乐后，将其放到目录res/raw中（必要时创建该目录）。例如，我将我选择并转换的音乐命名为a_guy_1_epicbuilduploop.ogg，并将其放在目录res/raw/中。你的Android Studio项目应类似于图6-1所示。

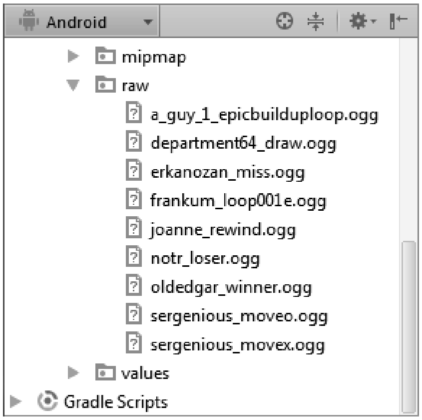


图6-1　将音乐和音效放在目录raw中

接下来，修改主活动，以播放和停止播放音乐。

ticTacToev4/src/main/java/org/example/tictactoe/MainActivity.java

```
import android.media.MediaPlayer;

public class MainActivity extends Activity {
   MediaPlayer mMediaPlayer;
   // ……
   @Override
   protected void onResume() {
      super.onResume();
      mMediaPlayer = MediaPlayer.create(this, R.raw.a_guy_1_epicbuilduploop);
      mMediaPlayer.setVolume(0.5f, 0.5f);
      mMediaPlayer.setLooping(true);
      mMediaPlayer.start();
   }

   @Override
   protected void onPause() {
      super.onPause();
      mMediaPlayer.stop();
      mMediaPlayer.reset();
      mMediaPlayer.release();
   }
}
```

① http://pragprog.com/book/eband4

方法onResume()在活动可见时被调用。我们在其中创建了一个多媒体播放器，并传入了目录res/raw中音频文件的资源id，然后稍微降低了音量，将播放模式设置为循环播放并启动播放器。

用户离开当前活动（进入下一个屏幕、退出程序或启动另一个应用）时，Android将调用onPause()，以停止播放音乐并释放播放器分配的所有资源。如果不这样做，将导致内存泄露，最终导致程序崩溃。

现在请启动这个游戏。在主屏幕显示期间，你将听到有一首歌曲在播放。播放一段时间后，你可能会感到厌烦，此时可按返回按钮或开始新游戏，让歌曲不再播放。

6.2　更换音乐

在游戏期间播放不同的歌曲可以起到不错的调剂作用。下面对GameActivity.java做这样的修改：添加游戏结束时要播放的声音。首先，需要声明一个多媒体播放器变量和一个后面要用到的Handler。

ticTacToev4/src/main/java/org/example/tictactoe/GameActivity.java

```
import android.media.MediaPlayer;
import android.os.Handler;

public class GameActivity extends Activity {
   private MediaPlayer mMediaPlayer;
   private Handler mHandler = new Handler();
   // ……
}
```

接下来，和前面一样，在方法onResume()和onPause()中分别开始播放和停止播放音乐。

ticTacToev4/src/main/java/org/example/tictactoe/GameActivity.java

```
@Override
protected void onResume() {
   super.onResume();
   mMediaPlayer = MediaPlayer.create(this, R.raw.frankum_loop001e);
   mMediaPlayer.setLooping(true);
   mMediaPlayer.start();
}

@Override
protected void onPause() {
   super.onPause();
   mHandler.removeCallbacks(null);
   mMediaPlayer.stop();
   mMediaPlayer.reset();
   mMediaPlayer.release();
   String gameData = mGameFragment.getState();
   getPreferences(MODE_PRIVATE).edit()
         .putString(PREF_RESTORE, gameData)
         .commit();
   Log.d("UT3", "state = " + gameData);
}
```

最后，修改方法reportWinner()，以便在游戏结束时根据获胜方来播放相应的轻松而有节奏的声音。

ticTacToev4/src/main/java/org/example/tictactoe/GameActivity.java

```
public void reportWinner(final Tile.Owner winner) {
   AlertDialog.Builder builder = new AlertDialog.Builder(this);
   if (mMediaPlayer != null && mMediaPlayer.isPlaying()) {
      mMediaPlayer.stop();
      mMediaPlayer.reset();
      mMediaPlayer.release();
   }
   builder.setMessage(getString(R.string.declare_winner, winner));
   builder.setCancelable(false);
   builder.setPositiveButton(R.string.ok_label,
         new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
               finish();
            }
         });
   final Dialog dialog = builder.create();
   mHandler.postDelayed(new Runnable() {
      @Override
      public void run() {
         mMediaPlayer = MediaPlayer.create(GameActivity.this,
               winner == Tile.Owner.X ? R.raw.oldedgar_winner
                     : winner == Tile.Owner.O ? R.raw.notr_loser
                     : R.raw.department64_draw
         );
         mMediaPlayer.start();
         dialog.show();
      }
   }, 500);

   mGameFragment.initGame(); // 将棋盘重置为初始状态
}
```

如果当前正在播放音乐，就调用方法MediaPlayer.stop()停止播放。接下来，设置一个Handler，以便在半秒（500毫秒）后运行一段代码。这段代码负责创建一个新的多媒体播放器，并传入要播放的声音的id。为此，我从freesound.org网站挑选了另外3个音频剪辑。

6.3 播放下棋声

观看电视节目或电影时，屏幕上出现图像的同时会伴有声音，这些声音通常不是拍摄时录制的，而是后期制作期间由拟音师添加的。在拟音期间，拟音师添加滴答声、呼呼声、脚步声、砰咚声、轰隆声等数千种声音。如果做得好，你根本感觉不到拟音师的存在。因此，下次请耐着性子看完演职人员名单，在其中寻找拟音师。

在这里，你将变身为拟音师，给井字游戏添加音效。具体地说，我们要让这个游戏在用户选择棋盘格或按重置游戏的按钮时发出声音。为了播放这些声音，我们将使用另一个类——`SoundPool`，它更适合用于播放简短的声音剪辑。不同于MediaPlayer，SoundPool可同时播放多种声音。它会跟踪所有的声音，并将它们混合在一起。

使用`new SoundPool()`来创建`SoundPool`实例。从Android 5.0（Lollipop）起，可以使用`SoundPool.Builder`类，但由于我们希望这个应用能够在更早的Android版本上运行，因此需要使用老旧的方式。如果编译器显示了警告消息，指出构造函数`SoundPool()`已被摒弃，不用管它就是了。不用担心，这个构造函数现在还管用。

ticTacToev4/src/main/java/org/example/tictactoe/GameFragment.java

```
import android.media.AudioManager;
import android.media.SoundPool;

public class GameFragment extends Fragment {
   private int mSoundX, mSoundO, mSoundMiss, mSoundRewind;
   private SoundPool mSoundPool;
   private float mVolume = 1f;
   // ……
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      // 在设备配置发生变化时保留这个片段
      setRetainInstance(true);
      initGame();
      mSoundPool = new SoundPool(3, AudioManager.STREAM_MUSIC, 0);
      mSoundX = mSoundPool.load(getActivity(), R.raw.sergenious_movex, 1);
      mSoundO = mSoundPool.load(getActivity(), R.raw.sergenious_moveo, 1);
      mSoundMiss = mSoundPool.load(getActivity(), R.raw.erkanozan_miss, 1);
      mSoundRewind = mSoundPool.load(getActivity(), R.raw.joanne_rewind, 1);
   }
```

方法`SoundPool.load()`接受3个参数，并返回加载的声音的id。这些参数分别是当前活动、原始声音文件的资源id以及声音的优先级。

这个方法并不会播放声。要播放声音，需要调用方法`SoundPool.play()`。将棋盘格的点击监听器（点击棋盘格时将执行的代码）修改成下面这样。

ticTacToev4/src/main/java/org/example/tictactoe/GameFragment.java

```
private void initViews(View rootView) {
         // ……
         inner.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
               if (isAvailable(smallTile)) {
                  mSoundPool.play(mSoundX, mVolume, mVolume, 1, 0, 1f);
                  makeMove(fLarge, fSmall);
                  think();
               } else {
```

```
                    mSoundPool.play(mSoundMiss, mVolume, mVolume, 1, 0, 1f);
                }
            }
        });
        // ……
    }
    private void think() {
                // ……
                switchTurns();
                mSoundPool.play(mSoundO, mVolume, mVolume, 1, 0, 1f);
                makeMove(move[0], move[1]);
                switchTurns();
                // ……
    }
```

首先播放X声音，指出用户已下棋。1秒钟后，播放O声音，指出计算机已下棋。如果用户选择的棋盘格不能下棋，就播放另一种声音，指出用户点击的棋盘格不对。

完成这些修改后，务必试玩一盘，并让各种声音播放出来，以测试它们。

顺便说一句，方法`SoundPool.play()`功能众多，它接受如下6个参数。

- `soundID`：函数`load()`返回的声音id。
- `leftVolume`：左声道的音量（取值范围为0.0~1.0）。
- `rightVolume`：右声道的音量（取值范围为0.0~1.0）。
- `priority`：流的优先级（0表示优先级最低）。
- `loop`：循环模式（0表示不循环，–1表示不断循环）。
- `rate`：播放速度（1.0表示正常播放速度，取值范围为0.5~2.0）。

请尝试修改这些参数，以获得不同的效果。

6.4 快速阅读指南

本章介绍了如何使用Android SDK播放音频剪辑。我们并没有讨论录音，因为大多数程序都不需要录音，但如果你的程序属于例外情况，请在在线文档[①]中查找`MediaRecorder`类。

第7章将结束井字游戏的编写工作，你将学习一些简单的动画技巧，它们可以让Android程序的娱乐性变得大不相同。如果你暂时不需要这样做，可直接跳到第8章，学习如何让应用能够在各种设备上运行。

① http://d.android.com/reference/android/media/MediaRecorder.html

第 7 章

添加动画

要让Android应用更有趣、更具娱乐性，有一种方法，那就是添加动画。如果制作得有品位，动画可以让烦闷的程序变得激动人心且充满活力。

不要让元素突然出现在屏幕上，而要让它们飞入或跳跃地进入；不要使用静态的背景，而要通过移动赋予背景以生机。对于必须有趣的游戏来说，尤其需要这样做。然而，过犹不及。使用动画是要改善用户体验的，而不是分散用户注意力。

在本章中，将在井字棋游戏中添加两个动画元素。首先，将在主菜单后面添加滚动的背景。然后，要让棋盘格在用户触摸时跳跃起来。

7.1　不断滚动的画卷

我们要在开始屏幕上显示一幅背景图像。它缓慢地向屏幕左上角滚动。用户看不到任何缝隙，动画非常平滑，如图7-1所示。



图7-1　如果你盯着看很长时间，它将看起来就像在不断移动

7.1.1　添加视图

要实现上述效果，需要创建一个自定义视图——ScrollingView，并让它覆盖游戏选择菜单后面的整个屏幕。编辑文件res/layout/activity_main.xml，将下面的元素作为第一个子元素插入到FrameLayout元素中。

ticTacToev5/src/main/res/layout/activity_main.xml

```
<org.example.tictactoe.ScrollingView
   android:id="@+id/main_background"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   app:scrollingDrawable="@drawable/xando_background"/>
```

上述代码有两个很重要的地方。首先，它引用的不是标准Android视图，而是org.example.tictactoe.ScrollingView，这个视图将稍后定义。其次，它设置了一个以前没有涉及过的属性app:scrollingDrawable。这是一个自定义属性，指定了滚动的图像。

7.1.2　定义自定义属性

注意，在前面的自定义属性中，包含前缀app:。这是一个XML命名空间，必须将其定义为FrameLayout标签的一个属性。为此，应在xmlns:android的定义后面添加如下属性。

```
xmlns:app="http://schemas.android.com/apk/res-auto"
```

这个特殊的命名空间会告诉Android，对于任何以app:打头的属性，都应在资源目录中查找其定义。创建文件res/values/attrs_scrolling_view.xml，并在其中添加如下定义。

ticTacToev5/src/main/res/values/attrs_scrolling_view.xml

```
<resources>
   <declare-styleable name="ScrollingView">
      <attr name="scrollingDrawable" format="color|reference" />
   </declare-styleable>
</resources>
```

使用什么样的文件名无关紧要，只要将它放在正确的目录中就行。这个文件指出，ScrollingView只有一个名为scrollingDrawable的属性，该属性的值可以是颜色，也可以是drawable引用。

7.1.3　背景信息

我们要滚动的背景是一个简单的重复位图。

ticTacToev5/src/main/res/drawable/xando_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<bitmap xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:src="@drawable/xando"
    android:tileMode="repeat" />
```

它引用了目录res/drawable-mdpi下的文件xando.png。你可以使用任何图像，只要它能够无缝地平铺即可（说起来容易做起来难）。网上也有很多这样的图像，但我是使用绘图程序手动创建了一个图像，你可以在本书配套网站[1]的ticTacToev5示例中找到该图像。

7.1.4　创建滚动视图

至此，余下的全部工作就是创建滚动视图。首先来创建这个类的轮廓。

ticTacToev5/src/main/java/org/example/tictactoe/ScrollingView.java

```
package org.example.tictactoe;

import android.content.Context;
import android.content.res.TypedArray;
import android.graphics.Canvas;
import android.graphics.drawable.Drawable;
import android.util.AttributeSet;
import android.view.View;

/**
 * 这个自定义视图负责绘制一幅不断滚动的背景图像
 */
public class ScrollingView extends View {
   private Drawable mBackground;
   private int mScrollPos;
}
```

创建包含属性的自定义视图时，Android要求必须添加如下构造函数。

ticTacToev5/src/main/java/org/example/tictactoe/ScrollingView.java

```
public ScrollingView(Context context) {
   super(context);
   init(null, 0);
}

public ScrollingView(Context context, AttributeSet attrs) {
   super(context, attrs);
   init(attrs, 0);
}

public ScrollingView(Context context, AttributeSet attrs, int defStyle) {
   super(context, attrs, defStyle);
   init(attrs, defStyle);
}
```

方法`init()`用于获取属性列表，并从中获取所需的属性，如下所示。

① http://pragprog.com/book/eband4

ticTacToev5/src/main/java/org/example/tictactoe/ScrollingView.java

```
private void init(AttributeSet attrs, int defStyle) {
   // 加载自定义视图的属性列表
   final TypedArray a = getContext().obtainStyledAttributes(
         attrs, R.styleable.ScrollingView, defStyle, 0);

   // 获取背景图像
   if (a.hasValue(R.styleable.ScrollingView_scrollingDrawable)) {
      mBackground = a.getDrawable(
            R.styleable.ScrollingView_scrollingDrawable);
      mBackground.setCallback(this);
   }

   // 不再需要属性列表
   a.recycle();
}
```

实际工作是由方法`onDraw()`完成的。

ticTacToev5/src/main/java/org/example/tictactoe/ScrollingView.java

```
@Override
protected void onDraw(Canvas canvas) {
   super.onDraw(canvas);

   // 获取视图的尺寸（不包括内边距）
   int contentWidth = getWidth();
   int contentHeight = getHeight();

   // 绘制背景
   if (mBackground != null) {
      // 让背景比实际需要的更大
      int max = Math.max(mBackground.getIntrinsicHeight(),
            mBackground.getIntrinsicWidth());
      mBackground.setBounds(0, 0, contentWidth * 4, contentHeight * 4);

      // 调整图像的绘制位置
      mScrollPos += 2;
      if (mScrollPos >= max) mScrollPos -= max;
      setTranslationX(-mScrollPos);
      setTranslationY(-mScrollPos);

      // 绘制图像并指出下次刷新时也应绘制它
      mBackground.draw(canvas);
      this.invalidate();
   }
}
```

这个方法的重要部分是`mBackground.draw()`，它负责在屏幕上绘制由X和O组成的背景。由于绘制前调用了`setTranslationX()`和`setTranslationY()`，这些X和O看起来像是在不断地移动。

`onDraw()`将被反复调用（每帧一次），这是因为在最后调用了`invalidate()`。每次调用

`onDraw()`时，滚动位置（`mScrollPos`）都加2，这将导致背景缓慢地向左上角移动。由于背景比屏幕大，用户永远都看不到其边缘，从而避免了缝隙的出现。

7.2　跳跃的棋盘格

在游戏屏幕中，若让X和O突然出现会显得不逼真，下面让棋盘格变得更有活力：让X和O滴到棋盘上，就像是凝胶做的那样。

7.2.1　动画原则

1981年，迪斯尼动画师Ollie Johnston和Frank Thomas出版了一部著名的著作*The Illusion of Life: Disney Animation* [JT95]，阐述了从20世纪30年代起动画师采用的动画制作流程。该书概述了12条基本的动画原则[①]，这些原则在今天依然适用。

这些原则对所有动画（而不仅仅是动画片）来说都至关重要。有鉴于此，下面将它们一一列出，然后演示如何在Android开发中遵循这些原则（有关这些原则的详细描述，请参阅在线资料）：

(1) 挤压与拉伸（Squash and stretch）

(2) 预备动作（Anticipation）

(3) 分镜（Staging）

(4) 逐帧绘制和使用关键帧（Straight ahead action and pose to pose）

(5) 随动和重叠动作（Follow through and overlapping action）

(6) 缓入缓出（Slow in and slow out）

(7) 弧线运动（Arc）

(8) 次要动作（Secondary action）

(9) 节奏（Timing）

(10) 夸张（Exaggeration）

(11) 扎实的绘画功力（Solid drawing）

(12) 吸引力（Appeal）

在Android中，动画是在XML文件中定义的。请在res目录中新建一个名为animator的子目录，再在这个子目录中新建一个名为tictactoe.xml的文件，并在其中添加如下内容。

① http://en.wikipedia.org/wiki/12_basic_principles_of_animation

ticTacToev5/src/main/res/animator/tictactoe.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
   <objectAnimator
      android:duration="500"
      android:interpolator="@android:interpolator/overshoot"
      android:propertyName="scaleX"
      android:valueFrom="2"
      android:valueTo="1" />
   <objectAnimator
      android:duration="700"
      android:interpolator="@android:interpolator/overshoot"
      android:propertyName="scaleY"
      android:valueFrom="2"
      android:valueTo="1" />

</set>
```

这个文件定义了一个动画集，该动画集同时也是影响对象多个属性的动画。在这里，影响的属性是对象的水平尺寸（宽度）和垂直尺寸（高度）。

对于这两个属性，我们都将其起始值设置得很夸张——正常值的2倍（原则10），然后逐渐将它们缩小到正常值。我们定义了起始帧和结束帧，并让计算机填充过渡帧（原则4）。

动画的节奏（原则9）是由属性`duration`设置的。注意到两个属性的变化持续时间稍有不同，从而创建了挤压和拉伸效果（原则1）。中间帧是由属性`interpolator`控制的，使用该属性创建了随动（泻落）效果（原则5）。通过将`interpolator`属性设置为不同的值，还可以创建缓入缓出效果（原则6）、预备动作效果（原则2）和弧线运动效果（原则7）。

7

7.2.2 走起

定义通用的动画后，需要在合适的时候将其应用于棋盘格。为此，在文件GameFragment.java中，要在方法`addAvailable()`开头调用`animate()`。

ticTacToev5/src/main/java/org/example/tictactoe/GameFragment.java

```
private void addAvailable(Tile tile) {
   tile.animate();
   mAvailable.add(tile);
}
```

这导致可着子的棋盘格将执行跳跃动画。

接下来，在`initViews()`内的方法`onClick()`开头，也调用`animate()`。

ticTacToev5/src/main/java/org/example/tictactoe/GameFragment.java

```
public void onClick(View view) {
   smallTile.animate();
   // ……
}
```

这将导致用户触摸的棋盘格跳跃起来。最后，在方法makeMove()中调用animate()，如下所示。

ticTacToev5/src/main/java/org/example/tictactoe/GameFragment.java

```
if (winner != oldWinner) {
   largeTile.animate();
   largeTile.setOwner(winner);
}
```

这将导致玩家X或O赢得小棋盘，小棋盘执行动画。如果不知道该在什么地方调用animate()，请参阅本书配套网站的示例代码。

下面来添加刚才调用的方法animate()。

7.2.3 观看跳跃的棋盘格

完成所有的准备工作后，不需要做太多的工作就能实际执行动画。为此，首先需要在文件Tile.java的开头添加两条import语句。

ticTacToev5/src/main/java/org/example/tictactoe/Tile.java

```
import android.animation.Animator;
import android.animation.AnimatorInflater;
```

然后，编写方法animate()，如下所示。

ticTacToev5/src/main/java/org/example/tictactoe/Tile.java

```
public void animate() {
   Animator anim = AnimatorInflater.loadAnimator(mGame.getActivity(),
         R.animator.tictactoe);
   if (getView() != null) {
      anim.setTarget(getView());
      anim.start();
   }
}
```

第1行用于加载前面定义的动画tictactoe.xml。接下来，如果有视图关联到棋盘格，就将该视图设置为动画的目标，再启动动画。

这就是运行动画所需的全部代码！

7.2.4 现状

祝贺你！下一章再做一些细微的调整，整个井字游戏示例就完成了。通过创建这个示例程序，你学习了创建基本的Android应用所需的全部知识，具体如下。

- 使用活动和片段创建应用。

- 使用XML创建用户界面。
- 响应用户输入。
- 添加声音和动画。

图7-2显示了最终完成的作品。

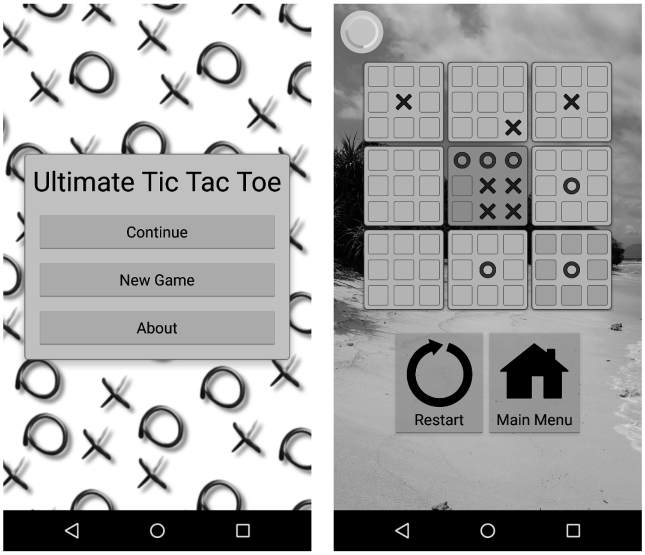


图 7-2

现在休息一会，来尝试玩玩这个游戏。将它介绍给你的朋友，然后大刀阔斧地改进它。别忘了，这个游戏以及本书所有示例的源代码都可在配套网站找到。

7.3 快速阅读指南

在本章中给井字游戏添加了两个动画效果：主屏幕滚动的背景以及游戏屏幕中跳跃的棋盘格。你在这里学到的技巧在很多情况下都将有用武之地。

下一章将探索如何让应用能够适应各种屏幕尺寸和设备。我们将妥善地处理这个井字游戏，让它在手机和平板电脑上看起来都会很漂亮，并能很好地运行。第9章将介绍如何将应用发布到Google Play Store，而从第10章开始，将转而探讨更高级的主题，如访问Internet以及使用基于位置的服务。

第三部分

创造性思维

第 8 章

编写一次，到处测试

当前，Android已经广泛用于各种手机、平板电脑、手表、电视等设备，这是福也是祸。对于消费者来说，这是福音，因为有形状、尺寸、价格各异的Android设备可供选择；但对于开发人员来说，这是麻烦，因为必须支持各种各样的设备。

雪上加霜的是，Android发展神速，导致Android设备运行的版本各不相同，非常分散。表8-1列出了已发布的所有Android版本，其中带星号的版本已不再使用。Android Platform Dashboard[①]提供了最新的图表，指出了运行各种Android版本的设备所占的比例。

`Build.VERSION_CODES`列出了所有的Android版本代号和API等级。

表8-1　Android版本代号和API等级

版　本	代　号	API 等级	发布时间	说　明
1.0*	BASE	1	2008 年 9 月	第一版
1.1*	BASE_1_1	2	2009 年 2 月	跑马灯效果、短信附件
1.5*	CUPCAKE	3	2009 年 5 月	小部件、虚拟键盘
1.6*	DONUT	4	2009 年 9 月	高密度屏幕和低密度屏幕
2.0*	ECLAIR	5	2009 年 11 月	Exchange 账户
2.0.1*	ECLAIR_0_1	6	2009 年 12 月	多点触摸
2.1*	ECLAIR_MR1	7	2010 年 1 月	动态壁纸
2.2*	FROYO	8	2010 年 5 月	SD 卡
2.3*	GINGERBREAD	9	2010 年 12 月	原生游戏编程
2.3.3	GINGERBREAD_MR1	10	2011 年 2 月	NFC
3.0*	HONEYCOMB	11	2011 年 2 月	片段、操作栏、Holo 主题
3.1*	HONEYCOMB_MR1	12	2011 年 5 月	USB API、游戏杆
3.2*	HONEYCOMB_MR2	13	2011 年 6 月	7 英寸屏幕
4.0*	ICE_CREAM_SANDWICH	14	2011 年 10 月	Roboto、统一的手机/平板电脑 UI

① http://d.android.com/resources/dashboard/platform-versions.html

（续）

版　本	代　号	API 等级	发布时间	说　明
4.0.3	ICE_CREAM_SANDWICH_MR1	15	2011 年 12 月	社交流 API
4.1	JELLY_BEAN	16	2012 年 6 月	Project Butter、systrace、可扩展的通知
4.2	JELLY_BEAN_MR1	17	2012 年 11 月	多用户、无线显示技术
4.3	JELLY_BEAN_MR2	18	2013 年 7 月	OpenGL ES 3.0、SELinux、权限设置（restricted profiles）
4.4	KITKAT	19	2013 年 10 月	Chromium WebView、沉浸模式
4.4W*	KITKAT_WATCH	20	2014 年 6 月	Android Wear
5.0	LOLLIPOP	21	2014 年 10 月	ART、Material design、Project Volat
5.1	LOLLIPOP_MR1	22	2015 年 3 月	多 SIM 卡、运营商服务

本章介绍如何在程序中支持多种Android版本和屏幕分辨率。第一步是测试。

8.1 启动模拟器

在本书的大多数地方，都会让你将应用的目标平台设置为Android 4.1（Jelly Bean）版。然而，这个建议存在一个小问题：程序可能无法在旧版本上运行。要确定这一点，唯一可靠的方式是测试。要在使用不同Android版本和屏幕尺寸的设备上测试程序，最佳的方式是使用模拟器，而不是购买市面上的每种Android手机和平板电脑。为此，你需要创建多个虚拟设备，它们的Android版本和屏幕尺寸各不相同。

8

8.1.1 模拟器反斗城

除1.3节创建的Nexus 5 AVD外，还建议你创建表8-2中所示的用于测试的虚拟设备。

表8-2 虚拟设备列表

名　称	目标平台	分辨率	密　度
Nexus 4	4.1	768×1280	xhdpi（超高点每英寸）
Nexus 5	5.1	1080×1920	xxhdpi（极高）
Nexus 7	4.2	1200×1920	xhdpi
Nexus 9	4.4	2048×1536	xhdpi

注意 新的Android版本推出后，务必将其纳入测试矩阵。要确定可将哪些旧版本排除在外，可参阅Android Platform Dashboard。

下面来创建运行Android 4.2的Nexus 4模拟器。在Android Studio中，单击AVD Manager按钮启

动AVD Manager，如图8-1所示。

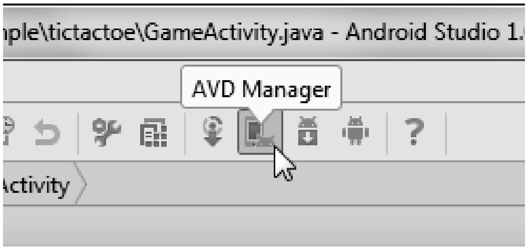


图　8-1

单击Create Virtual Device按钮，再从Phone类别中选择Nexus 4，然后单击Next按钮，如图8-2所示。

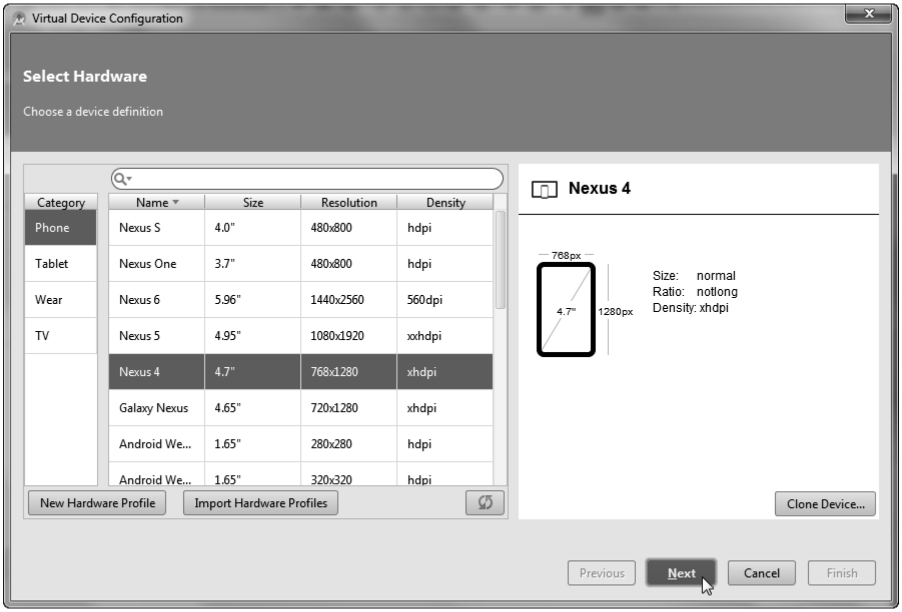


图　8-2

现在，选择一个系统映像，如图8-3所示。我们要让Nexus 4运行Android 4.1（Jelly Bean），因此选择其x86版（虽然大多数设备使用的都是ARM处理器，但应始终使用Intel x86模拟器，因为其速度要比ARM模拟器快几倍）。

如果所需的映像还未下载，单击Download链接下载它。

接下来，单击Next按钮，然后单击Finish按钮。创建前面列出的其他模拟器，直到看到图8-4所示的列表。

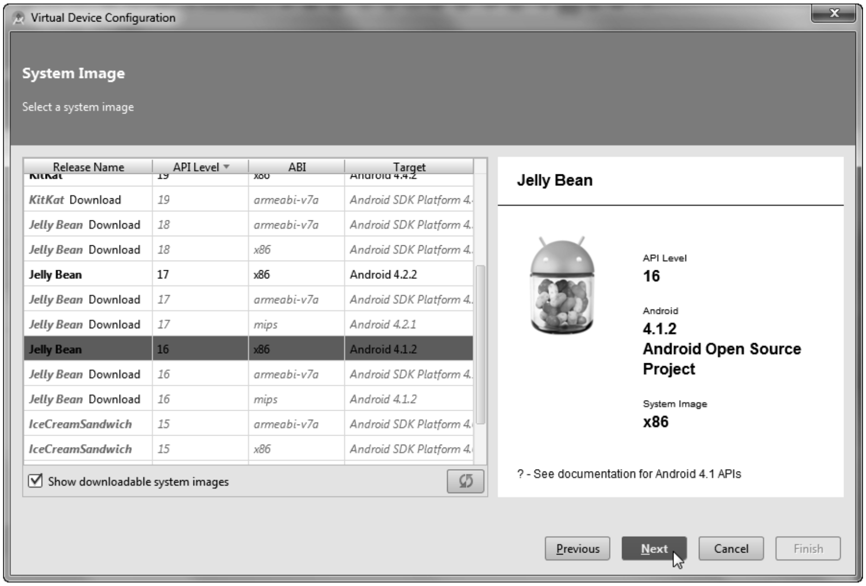


图 8-3

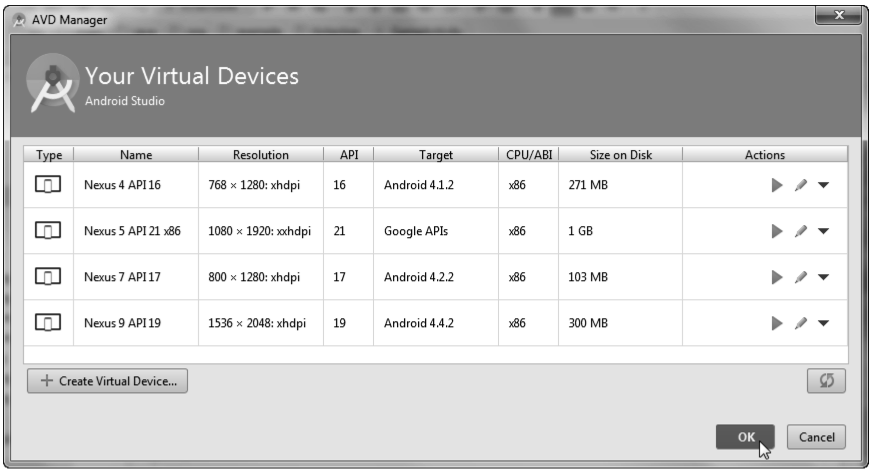


图 8-4

8.1.2 测试策略

开发时可使用Nexus 5 AVD，并在发布应用前在其他AVD中进行测试。另外，别忘了在纵向和横向模式下进行测试。在模拟器中，要在横向和纵向模式下切换，可按Ctrl+F11键，也可（在NumLock处于关闭状态时）使用数字小键盘中的7或9。

除非你的台式机配置极高，否则可能无法同时运行这些AVD。实际上，我发现每次只运行一个的效果最佳。

如果当前正在运行Nexus 5模拟器，请关闭它，再启动模拟器Nexus 4：在AVD Manager中，单击Actions栏中的播放图标。模拟器显示主屏幕后，如果屏幕被锁定，将其解锁。下面介绍一些可能出现问题的情况。

8.2　测试程序

现在运行本书前面创建的井字游戏程序，它将出现在Nexus 4模拟器中。遗憾的是，它看起来并不是太好，如图8-5所示（请不要太难过）。

点击New Game按钮开始游戏。一切看起来都不错，但下棋后，所有的空棋盘格都变成了黑色的，如图8-6所示。



图　8-5

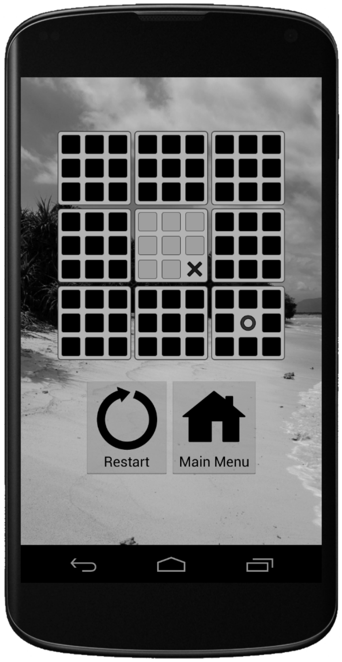


图　8-6

如何修复这样的问题呢？

一种技巧是，尝试在越来越新的Android版本中运行程序，直到找到没有问题的版本为止。如果你这样做了，将发现在Android 4.3中，模糊的问题消失了，而在Android 4.2中，棋盘格为黑色的问题也没有了。然后，你就可以查看这两个版本的发行说明，看看它们有哪些变化。你甚至

可以查看源代码。然而，这样做需要很长的时间。

另一种方法是，尝试以不同的方式重新实现代码，看看效果是否更好。这种方法虽然不那么科学，但常常能够在更短的时间内奏效。我将这种方法称为Fonzie——情景喜剧*Happy Days*中的一个角色。在这部情景喜剧中，自动唱机经常坏。每到这个时候，Fonzie都会走过去踢一脚或抖一抖，然后自动唱机就好了。同样，你可对代码做细微的修改，看看问题是不是消失了。

在这里，我们结合使用了这两种方法来修复问题。首先，我找出这些问题消失了的版本。但在发行说明中没有找到明显的线索，指出什么变化是罪魁祸首。因此，我开始修改代码。在做了一番调整之后我发现，对于ScrollingView.java中的如下代码：

ticTacToev5/src/main/java/org/example/tictactoe/ScrollingView.java

```
// 调整图像的绘制位置
mScrollPos += 2;
if (mScrollPos >= max) mScrollPos -= max;
setTranslationX(-mScrollPos);
setTranslationY(-mScrollPos);
```

如果将其修改成下面这样：

ticTacToev6/src/main/java/org/example/tictactoe/ScrollingView.java

```
// 调整图像的绘制位置
mScrollPos += 2;
if (mScrollPos >= max) mScrollPos -= max;
canvas.translate(-mScrollPos, -mScrollPos);
```

就可修复模糊的问题。接下来，我研究tile_empty.xml的定义。

ticTacToev5/src/main/res/drawable/tile_empty.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
       android:shape="rectangle">
   <stroke
      android:width="@dimen/stroke_width"
      android:color="@color/dark_border_color"/>
   <corners android:radius="@dimen/corner_radius"/>
</shape>
```

发现如果像下面这样给形状指定填充色，就可以修复棋盘格为黑色的问题。

ticTacToev6/src/main/res/drawable/tile_empty.xml

```
<solid android:color="@color/gray_color"/>
```

如果什么办法都不管用

对于棘手的问题，可进行版本检查。例如，可以使用代码检查设备运行的是否是较新的Android版本。如果不是，就完全禁用滚动效果。

```
// 避免运行较旧版本不支持的代码
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
    // 滚动代码……
}
```

你可以根据Android版本使用不同的资源，这将在8.3节中进行更详细的讨论。

最后，如果实在找不到解决问题的简单办法，可以修改应用针对的最低版本。但别忘了，这样做会缩小目标用户的范围。

8.3　大小屏幕通吃

为了让应用在尽可能多的Android设备上看起来都是最漂亮的，必须支持不同的屏幕尺寸、分辨率和像素密度。Android会尽力缩放用户界面以适应设备，但它并非什么时候都做得很好。要获悉结果，唯一的办法你可能猜到了，那就是测试。为了确保程序在尺寸最常见的设备上能够正常运行，请使用8.1节推荐的模拟器类型。

作为测试，请尝试在Nexus 9模拟器中运行井字游戏。你将发现，主活动中的菜单和游戏活动中的棋盘都太小了。这是因为，所有的尺寸都是以dp为单位指定的，而平板电脑的屏幕比手机要大得多（即水平和垂直dp数更多）。有鉴于此，在平板电脑上需要增大尺寸。

Google Wear、TV和Auto

在手机和平板电脑上运行的程序，也可以在手表（Google Wear）、电视（Google TV）和汽车（Google Auto）上运行。为此，需要首先在模拟器中运行它们，看看其外观如何，然后再调整布局和尺寸。手表、电视和汽车大多不支持触摸屏，因此必须对应用进行修改，以便能够使用其他输入方法。

对于Wear应用，有两种选择：直接在手表上运行；在手机上运行，并在手表上显示通知。有关Wear应用的更详细信息，请参阅Google网站的培训教程Building Apps for Wearables[1]。

对于TV应用，需要针对“后仰”体验进行专门设计，因为屏幕通常离用户有几英尺。培训教程Building Apps for TV[2]提供了很多极有帮助的小贴士。

汽车领域为Android提供了新的成长机会。有关这方面的更详细信息，请参阅在线文档中的Building Apps for Auto[3]部分。

1. http://d.android.com/training/building-wearables.html
2. http://d.android.com/training/tv
3. http://d.android.com/training/auto

8.3.1 指定替代资源

要针对特定的Android设备配置调整布局或图像，可在资源名中使用后缀。

例如，对于用于超高密度、高密度和中密度屏幕的图像，可将它们分别放在目录res/drawable-xhdpi、res/drawable-hdpi和res/drawable-mdpi中。在本书的所有示例中，对于将显示在主屏幕上的程序图标都做了这样的处理。对于独立于密度（即不应缩放）的图形，将其放在目录res/drawable-nodpi中。

好在我们将所有尺寸都放在了一个地方——目录values下的文件dimens.xml中，因此只需创建这个文件的不同版本，并将其放在名称中包含不同屏幕尺寸的目录中即可。在项目窗口中，这些限定符将出现在目录名（Project模式）或指示标签（Android模式）中。

表8-3按优先级列出了有效的目录名限定符（有关Android如何查找最匹配目录的详细说明，请参阅在线文档[①]）。

表8-3 有效的目录名限定符

限定符	值
MCC 和 MNC	移动国家代码和可选的移动网络代码。不推荐使用这种限定符
语言和地区	两个字母表示的语言以及两个字母表示的地区代码（在前面加上小写字母 r）。例如，fr、en-rUS、es-rES
排版方向	应用的排版方向，应为 ldltr（从左到右，默认设置）或 ldrtl（从右到左）
最小宽度	可用屏幕尺寸中较小的那个，如 sw320dp、sw600dp、sw720dp
宽度	以 dp 为单位的最小屏幕宽度，屏幕宽度超过指定值后才使用相应的资源，如 w720dp、w1024dp
高度	以 dp 为单位的最小屏幕高度，屏幕高度超过指定值后才使用相应的资源，如 h720dp、h1024dp
屏幕尺寸	small、normal、large、xlarge
宽屏幕/高屏幕	long、notlong
屏幕朝向	port、land
UI 模式	car、desk、television、appliance、watch
夜间模式	night、notnight
屏幕像素密度	ldpi、mdpi、hdpi、xhdpi、xxhdpi、xxxhdpi、nodpi、tvpi
触摸屏类型	notouch、finger
键盘是否可用	keysexposed、keyshidden、keyssoft
键盘类型	nokeys、qwerty、12key
可否导航	navexposed、navhidden
导航类型	nonav、dpad、trackball、wheel
SKD 版本	设备支持的 API 等级，在前面加上小写字母 v，如 v16、v21

① http://d.android.com/guide/topics/resources/providing-resources.html#BestMatch

要使用多个限定符，只需使用连字符（-）将它们连接起来即可。例如，目录res/drawable-fr-land-hdpi包含如下情形下使用的图片：法文、横向模式和高密度屏幕。

8.3.2　调整游戏界面的大小

以井字游戏为例，来看看需要创建哪些文件。

values/dimens.xml中的基本设置将用于最小的手机。右击目录res并选择New➤Directory，以创建其他4个目录：values-sw360dp、values-sw600dp、values-sw720dp和values-w820dp（注意到除最后一个目录外，使用的都是“最小宽度”限定符。另外，最后一个目录可能已经是存在的）。项目窗口处于Android模式时，其中不会显示这些目录，但当你在values目录下添加文件时，会要求你指定限定符。将文件values/dimens.xml复制到这些目录中，并只重写要修改的值。我所使用的文件如下。

values-sw360dp（手机）：

ticTacToev6/src/main/res/values-sw360dp/dimens.xml

```
<resources>
   <dimen name="tile_size">35dp</dimen>
</resources>
```

values-sw600dp（小型平板电脑）：

ticTacToev6/src/main/res/values-sw600dp/dimens.xml

```
<resources>
   <!-- 默认屏幕边距，根据Android Design指导原则 -->
   <dimen name="activity_horizontal_margin">16dp</dimen>
   <dimen name="activity_vertical_margin">16dp</dimen>

   <dimen name="tile_size">50dp</dimen>
   <dimen name="tile_padding">5dp</dimen>
   <dimen name="small_board_padding">4dp</dimen>
   <dimen name="small_board_margin">4dp</dimen>
</resources>
```

values-sw720dp（中型平板电脑）：

ticTacToev6/src/main/res/values-sw720dp/dimens.xml

```
<resources>
   <dimen name="tile_size">70dp</dimen>
</resources>
```

values-w820dp（宽屏）：

ticTacToev6/src/main/res/values-w820dp/dimens.xml

```
<resources>
   <dimen name="activity_horizontal_margin">64dp</dimen>
</resources>
```

要确定正确的值，需要做一些错误测试。好在Android Studio提供了使这种工作实施起来更容易的功能。

8.3.3　预览

使用Android Studio的“预览”（Preview）窗口，可同时查看修改对多种屏幕尺寸的影响。编辑布局文件（如res/layout/activity_game.xml）时，将窗口调整到尽可能大，再选择XML右边的“预览”窗口中的Configuration图标。如果没有看到“预览”窗口，可选择菜单View➤Tool Windows➤Preview来打开它。选择Preview All Screen Sizes，以并排显示多种手机和平板电脑（包括纵向模式和横向模式），如图8-7所示。

接下来，创建并编辑dimens.xml文件，每次修改后都切换到activity_game.xml的“预览”窗口。不断地重复这种做法，直到得到所有屏幕都适合的效果。最后，在每个模拟器和实际设备上运行这个游戏，并对其进行测试。请务必旋转屏幕，在横向和纵向模式下都进行测试。

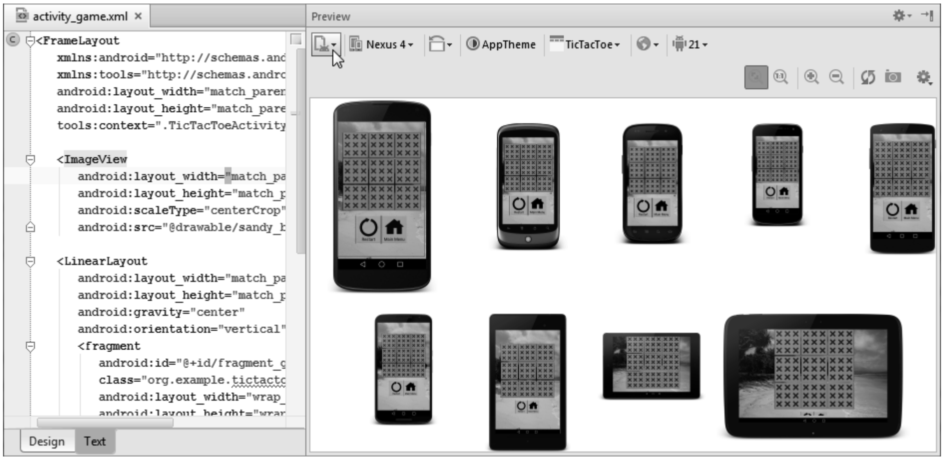


图　8-7

8.3.4　使用样式

对这个游戏还需做一项调整。Android 5.0（Lollipop）引入了新的Material主题，改变了UI元素的外观和行为。在Android 5和更高版本中运行时，要使用这种最新的外观，否则应用看起来会显得很过时。为此，创建一个名为res/values-v21的目录，并在其中添加文件styles.xml（21是Android 5.0的API等级）。

ticTacToev6/src/main/res/values-v21/styles.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
```

```
    <!-- 应用的基本样式 -->
    <style name="AppTheme"
           parent="android:Theme.Material.Light.NoActionBar.Fullscreen">
    </style>
</resources>
```

这样就修改了应用样式，将主题Material应用于全屏应用。要查看这样做后有何变化，可进入“预览”窗口，并从“配置”下拉列表中选择Preview Android Versions。

8.4　快速阅读指南

要支持运行不同Android版本且屏幕尺寸各不相同的硬件设备，这并不那么容易。本章介绍了最常见的问题和解决方案，为开展上述工作奠定了基础。如果你要更深入地了解这方面内容，建议阅读Android网站卓越的最佳实践文档“Supporting Multiple Screens”①。

经过一番艰苦努力的工作，你终于编写好了应用。接下来是有趣的部分：让人使用它。下一章介绍如何将应用发布到Google Play Store。

① http://d.android.com/guide/practices/screens_support.html

第 9 章

发布到Google Play Store

在本书前面，你只是在模拟器中运行创建的软件或将其下载到自己的Android手机。你为下一步做好了准备吗？通过发布到Google Play Store①，你的应用可供数以百万计的其他Android用户使用。本章就来告诉你如何实现这一目标。

9.1 准备工作

当然，要将应用发布到Play Store，你得先编写它。关于如何编写应用请参阅本书的其他部分。然而，仅编写代码还不够，你的程序还应该质量上乘，没有bug，并与尽可能多的设备兼容。下面的一些小贴士可助你一臂之力。

- 向别人展示前至少在一台实际设备上先进行测试。就算将其他小贴士都忘了，也别忘了这一条。
- 确保程序简单而精致。让程序只做一件事情，并做好这件事，而不要什么都做，什么都做不好。
- 选择你能够长期接受的Java包名，如com.yourcompany.prog-name。Android将AndroidManifest.xml指定的包名用作应用的主标识符。任何两个程序的包名都不能相同。将程序上传到Play Store后，除非将其删除，要求所有用户卸载，然后重新发布，否则无法修改包名。
- 在文件AndroidManifest.xml中，给属性`android:versionCode`和`android:versionName`设置有意义的值②。在命名方案中，给未来的更新留出空间。
- 遵循Android最佳实践③，如，设计时必须考虑性能、响应速度和无缝性。
- 遵循用户界面设计指南④，如图标设计、菜单设计、妥善地使用“后退”按钮。

9

① http://play.google.com/store

② http://d.android.com/guide/publishing/versioning.html

③ http://d.android.com/guide/practices

④ http://d.android.com/design

Android程序员面临的最严峻的挑战之一是，确保程序与多种设备兼容。你必须面对的一个问题是，用户设备的屏幕尺寸是各不相同的。有关这方面的建议，请参阅第8章。

第一印象和兼容性虽然重要，但必须在如下两方面取得平衡，即在对程序精雕细刻使其臻于完美的同时，确保能够按时发布它。若你认为已准备就绪，接下来就需要进行签名。

9.2　签名

必须将应用打包成一个.apk文件并使用数字证书进行签名，这样Android才会考虑运行它。对于模拟器和你的测试设备来说如此，对于要发布到Play Store的程序来说更是如此。

你可能会说，"等等，我在本书前面从未进行过打包或签名呀"。实际上，你这样做了。Android SDK工具使用证书悄悄地进行了编译和签名，而该证书是Google根据已知的别名和密码创建的。由于密码是已知的，因此不会提示你输入它，你甚至根本意识不到它的存在。然而，对于要发布到Play Store的应用，不能使用调试证书进行签名，因此你不能再依靠这个证书，而必须要创建自己的证书。

创建证书的方式有两种：使用标准Java命令`keytool`和`jarsigner`手动创建[1]，使用Android Studio自动创建。这里只介绍第二种。

从屏幕顶部的菜单中，选择Build➤Generate Signed APK。然后，选择要对其进行签名的模块，再单击Next按钮，如图9-1所示。

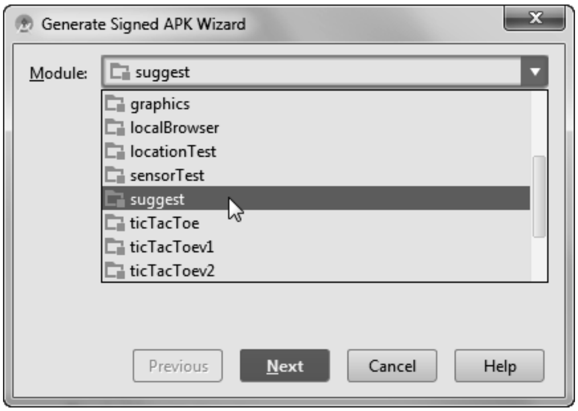


图　9-1

如果这是你首次给应用签名，请在下一个屏幕中单击Create new按钮。Android Studio将要求你指定密钥存储区路径，还有创建密钥存储区及其包含的密钥所需的一系列信息，如图9-2所示。

① http://d.android.com/tools/publishing/app-signing.html

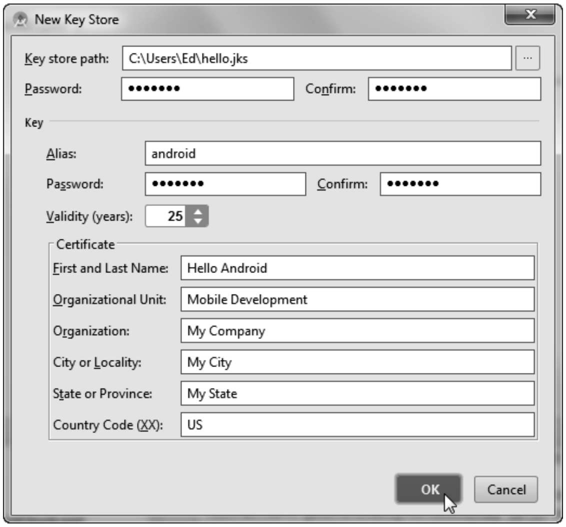


图9-2　千万不要把密码忘了

提示　请将密码记录下来并放在安全的地方。如果忘记了密码，密钥将毫无用处，而且你还无法更新应用。

单击OK按钮回到前一个屏幕，其中填写了所有的值，如图9-3所示。



图9-3　这个文件千万不能丢

对于你编写的所有应用的所有版本，都应使用相同的密钥。此外，还应采取合适的预防措施，防止私钥落入不法之徒手中。可以将其备份到Google网盘或其他安全的地方，以防万一。

单击Next按钮，再单击Finish按钮以启动签名过程。签名完成后，Android Studio将显示一条消息，如图9-4所示。



图　9-4

注意　如果你使用了Google Maps API，还需要从Google获取新的Maps API密钥，因为它与你的数字证书相关联。创建签名后的应用，按在线说明获取Maps API密钥[①]，修改AndroidManifest.xml文件以使用这个新密钥，然后再次对程序进行签名。

完成上述工作后，将得到一个可测试和发布的APK文件。

9.3　测试

Google建议发布签名的发行版之前，至少要在一台实际设备上进行测试（必须承认，我很少这样做）。在设备上安装发行版比安装调试版会复杂些。

Android Studio提供了一项名为build variants的功能。你可以使用它来将发行版安装到设备上。但它不能与Signed APK Wizard协同工作，因此我们将使用命令行工具来完成这项工作。

首先，打开一个命令窗口（也叫shell或终端窗口），并切换到前面创建的APK签名文件所在的目录，如图9-5所示。

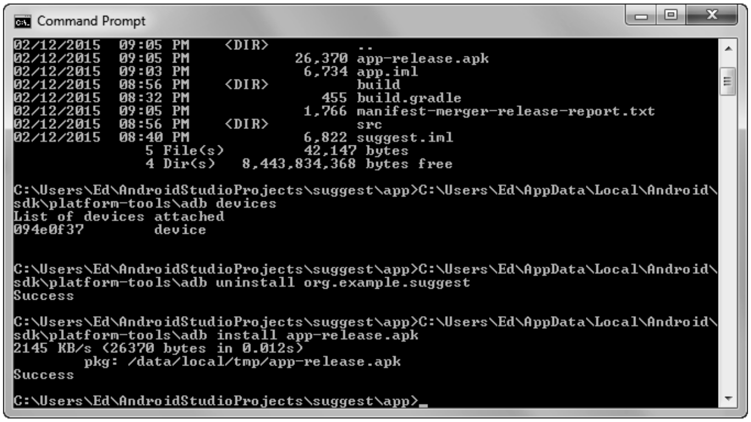


图9-5　80后希望能够使用命令行工具

① https://developers.google.com/maps/documentation/android

将设备插入计算机，再运行命令`adb devices`，并确定能够看到该设备。接下来，运行命令`adb uninstall your.package.name`（请将`your.package.name`替换为AndroidManifest.xml文件中指定的包名），将应用的调试版从设备中删除。最后，运行命令`adb install app-release.apk`（将`app-release.apk`替换为你的APK文件的名称），将程序复制到你的手机或平板电脑。

复制应用后，在你的设备中找到应用列表，再轻按该应用的图标以启动它。通过测试核实它能正确地运行。

9.4 发布

Google Play Store是Google提供的一项服务，可以使用它来发布所有的程序。为此，首先需要前往Developer Console[①]注册成为注册的开发人员。这需要支付少量的注册费。

如果你的程序是收费的，还需向一家支付处理商（payment processor）注册（上述网站会告诉你如何注册）。

现在可以上传了。单击链接Add new application，并填写打开的表单。下面是一些填写技巧。

- 除非有充分的理由，否则将Locations设置为All Current and Future Countries。鉴于不断有国家加入，这样设置能够让你的应用在所有国家都可以下载到。
- 在“联系人信息”（Contact Information）中，不要填写电话号码。所有Play Store用户都能看到这项信息，他们遇到问题时就会拨打这个号码。当然，如果你有专用的电话支持人员和号码，就不需要注意这一点了。

下面来演示发布第11章的示例应用suggest时，我是如何填写这个表单的。

```
Upload new APK: (select Choose File and Upload)
Language: English
Title (English): Suggest
Short description (English):
   Web service example generates amazingly accurate search suggestions.
Description (English):
   Suggest matches a partial word or phrase you enter and comes
   up with a list of possible completions. Select one of the
   results to do a search on that item.

   The results are amazingly accurate, and are refined based on
   your location and searches other people are doing right now.

   This program is a sample from an upcoming edition of "Hello,
   Android" by Ed Burnette, published by the Pragmatic Programmers.
   It demonstrates calling web services, multi-threading, XML
   Parsing, and the Google Suggest API.
```

① http://play.google.com/apps/publish

```
Application Type: Applications
Category: Libraries &amp; Demo

Content Rating: Everyone
Website: http://pragprog.com/book/eband4
Email: (my email address)
Phone: (blank)

Privacy Policy: (blank/not submitting a privacy policy)
Price: Free
Distribute in these countries: All
```

单击Publish按钮后，应用将出现在Google Play Store，在所有适用的设备上都能看到该应用。Google的批准流程很简单，只需几小时（而不是几周）即可完成，而且对程序的功能没有限制。准确地说是几乎没有限制，因为你必须遵循Google Play开发人员计划策略（Developer Program Policies）[1]，否则你的应用将从Google Play Store删除。其中一项规定是，应用的内容不得是非法、淫秽、令人反感或引发暴力、不适合18岁以下的人员观看的。另外，还必须遵守授权运营商的服务条款。确实有一些应用因用户或运营商的投诉而被删除掉了，但只要根据常识做出判断，就不用担心这种问题。

下一节介绍如何更新已发布的应用。

9.5　更新

假设应用发布到Google Play Store以后，你需要修改它。最简单的修改就是修改程序的元数据，即标题、描述、定价以及你在9.4节填写的所有信息。要修改非代码信息，只需在Developer Console的列表中选择程序，执行修改，再单击Submit Update即可。

如果有新的代码版本呢？没问题，你也可以在这个页面中上传它。但这样做之前，要花点时间确保修改了AndroidManifest.xml文件中的两个版本号，即每次上传时，都将`android:versionCode`的值加1（例如，从1改为2），并将属性`android:versionName`中人类可读的版本号增加合适的值（例如，修复小bug后，将该版本号从1.0.0改为1.0.1）。版本号正确无误，并重新打包和签名后，在Developer Console的APK部分单击按钮Upload new APK，然后单击Browse按钮，找到新的.apk文件，再单击OK按钮即可将其发送到服务器。

建议尽可能只是偶尔进行更新。请牢记下面两个矛盾体。

- 频繁更新会让用户高兴，因为这让他们认为你在提供支持，并倾听了他们的建议。
- 频繁更新也会让用户讨厌，因为他们将遭受更新通知的狂轰滥炸，进而可能将频繁骚扰他们的应用卸载。

① http://play.google.com/about/developer-content-policy.html

9.6 小贴士

下面是一些Google Play Store小贴士。这是我根据自己发布程序的经验总结出来的。

- 可以将收费的应用改成免费的，但不能将免费应用改成收费的。只要有可能，就应提供程序的免费（轻量级）版和收费（专业）版。千万不要删除免费版的任何功能，否则抗议将如暴风骤雨般袭来。
- 不能使用发布收费应用的账户购买该应用。
- 阅读用户留下的所有评论。对于特别粗鲁或低俗的垃圾评论，要毫不犹豫地报告。确保评论区域干净整洁，以接受有用的反馈（无论是赞扬还是批评）。
- 不要气馁。用户可能会毫不留情，发布气愤的评论时尤其如此。厚脸皮和幽默感是无价之宝。
- 除Google Play Store之外，还有其他一些应用商店。除发布到Google Play Store外，同时将应用发布到Amazon Appstore和其他深受欢迎的应用商店，这样将有更多的潜在用户看到它。

9.7 快速阅读指南

至此，你已经具备了编写成功Android应用并将其发布到Google Play Store所需的全部技能。你可以就此止步，也可以继续往下阅读，学习如何使用Android联网功能（第10章）和Google Play服务（第12章）。

第四部分

进 阶

第 10 章

联　网

接下来的几章介绍较高级的主题，如网络访问和基于位置的服务。即便不使用这些特性，也能编写很多有用的应用，但如果使用它们，只需做很少的工作就能添加大量功能，从而提高程序的价值。

用手机做什么呢？除了打电话，越来越多的人将手机用作移动网络设备。手机和平板电脑已经超越了台式机，成为联网的首要方式①。

Android手机装备精良，非常适合用于连接到移动网络。首先，Android提供了功能齐备的Web浏览器。该浏览器基于开源项目Chromium②。这是台式机浏览器Google Chrome使用的浏览器技术。Chromium是Apple iPhone、iPad和台式机浏览器Safari使用的WebKit③的一个分支（fork）。

其次，Android能够让你将该浏览器作为一个组件嵌入到应用中。最后，Android能够让你的程序访问标准网络服务，如TCP/IP套接字，以便让你能够使用Google、Yahoo和Amazon提供的Web服务以及Internet上的众多其他资源（包括你自己编写的资源）。

在本章中，你将通过3个示例学习如何集成Android的Web浏览功能。

- BrowserIntent：学习如何使用Android意图打开外部Web浏览器。
- BrowserView：了解如何在应用中嵌入浏览器。
- LocalBrowser：学习嵌入式WebView中的JavaScript和Android程序中的Java代码如何通信。

第11章将介绍如何在Android程序中使用基于云的服务和资源。

10.1　使用意图浏览网页

使用Android联网API可做的最简单的事情是打开一个浏览器，并在其中显示指定的网页。你可能想在程序中提供一个链接，让用户能够访问你的主页或基于服务器的应用（如预定系统）。

① http://sewat.ch/2353616

② http://www.chromium.org/Home

③ http://webkit.org

在Android中，只需编写3行代码并做一些设置工作，就能完成这种任务。

为了证明这一点，下面来编写一个新示例——BrowserIntent。这个示例程序包含一个文本框（edit field）和一个Go按钮。用户在文本框中输入URL后点击Go按钮，就可以打开浏览器并在其中显示URL指定的网页，如图10-1所示。首先，新建一个项目，在新建项目向导（New Project wizard）中做如下设置。

- ❑ 应用名：BrowserIntent
- ❑ 公司域名：example.org
- ❑ 尺寸：Phone and Tablet
- ❑ 最低SDK：API 16: Android 4.1（Jelly Bean）
- ❑ 添加活动：Blank Activity
- ❑ 活动名：MainActivity
- ❑ 布局名：activity_main
- ❑ 标题：Browser Intent

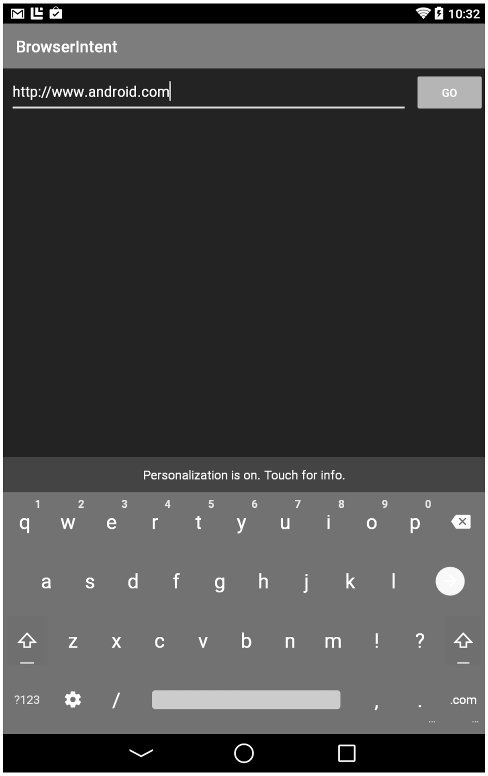


图 10-1

创建基本程序后，将布局文件（res/layout/activity_main.xml）修改成如下内容。

browserIntent/src/main/res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="horizontal"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">
   <EditText
      android:id="@+id/url_field"
      android:layout_width="0dip"
      android:layout_height="wrap_content"
      android:layout_weight="1.0"
      android:lines="1"
      android:inputType="textUri"
      android:imeOptions="actionGo" />

   <Button
      android:id="@+id/go_button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/go_button" />
</LinearLayout>
```

这里定义了前面所说的两个控件：一个EditText和一个Button。

在EditText的定义中，android:layout_weight="1.0"让这个文本区域填满按钮左边的全部水平空间，android:lines="1"将这个控件的高度限制为1行。请注意，这里限制的不是用户可在其中输入的文本量，而是显示的文本量。

android:inputType="textUri"和android:imeOptions="actionGo"指定了软键盘的外观，它们让Android将标准键盘替换为这样的键盘，即只包含方便输入网址的按钮。有关输入选项的更详细信息，请参阅TextView类的在线文档[①]。

与往常一样，供用户阅读的文本应放在资源文件res/values/strings.xml中。

browserIntent/src/main/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name">BrowserIntent</string>
   <string name="action_settings">Settings</string>
   <string name="go_button">Go</string>
</resources>
```

接下来，需要编写MainActivity类的方法onCreate()。我们将在这里创建用户界面并指定所有的行为。如果你不想输入这些代码，可以从本书配套网站[②]下载完整的源代码。

① http://d.android.com/reference/android/widget/TextView.html

② http://pragprog.com/book/eband4

browserIntent/src/main/java/org/example/browserintent/MainActivity.java

```
Line 1 package org.example.browserintent;
     -
     - import android.app.Activity;
     - import android.content.Intent;
     5 import android.net.Uri;
     - import android.os.Bundle;
     - import android.view.KeyEvent;
     - import android.view.View;
     - import android.view.View.OnClickListener;
    10 import android.view.inputmethod.EditorInfo;
     - import android.view.inputmethod.InputMethodManager;
     - import android.widget.Button;
     - import android.widget.EditText;
     - import android.widget.TextView;
    15 import android.widget.TextView.OnEditorActionListener;
     - public class MainActivity extends Activity {
     -    private EditText urlText;
     -    private Button goButton;
     -    @Override
    20    public void onCreate(Bundle savedInstanceState) {
     -       super.onCreate(savedInstanceState);
     -       setContentView(R.layout.activity_main);
     -       // 获取指向各个用户界面元素的句柄
     -       urlText = (EditText) findViewById(R.id.url_field);
    25       goButton = (Button) findViewById(R.id.go_button);
     -       // 设置事件处理程序
     -       goButton.setOnClickListener(new OnClickListener() {
     -          public void onClick(View view) {
     -             openBrowser();
    30          }
     -       });
     -       urlText.setOnEditorActionListener(new OnEditorActionListener() {
     -          public boolean onEditorAction(TextView v, int actionId,
     -                KeyEvent event) {
    35             if (actionId == EditorInfo.IME_ACTION_GO) {
     -                openBrowser();
     -                InputMethodManager imm = (InputMethodManager)
     -                      getSystemService(INPUT_METHOD_SERVICE);
     -                imm.hideSoftInputFromWindow(v.getWindowToken(), 0);
    40                return true;
     -             }
     -             return false;
     -          }
     -       });
    45    }
     - }
```

在方法`onCreate()`中，第22行调用了`setContentView()`来加载布局资源定义的视图，第24~25行调用`findViewById()`来获取指向两个用户界面控件的句柄。

第27行让Android在用户按下Go按钮（触摸该按钮或导航到该按钮后按中间的D-pad按钮）时运行一些代码——稍后将定义的方法`openBrowser()`。

如果用户输入网址后，按下了软键盘中的“链接”按钮或物理键盘中的回车键，也应该打开浏览器。为此，我们定义了另一个始于第32行的监听器，它在用户在文本框中执行操作时被调用。如果用户按下的是“链接”按钮，就调用方法`openBrowser()`来打开浏览器；否则就返回`false`，让文本框以正常的方式处理事件。

接下来是你期待已久的部分：方法`openBrowser()`。像前面所说的一样，它只包含3行代码。

browserIntent/src/main/java/org/example/browserintent/MainActivity.java

```
/** 打开浏览器并浏览到文本框中指定的URL */
private void openBrowser() {
   Uri uri = Uri.parse(urlText.getText().toString());
   Intent intent = new Intent(Intent.ACTION_VIEW, uri);
   startActivity(intent);
}
```

第1行将用户输入的网址转换为字符串（如http://www.android.com），再将其转换为统一资源标识符（URI）。

注意　输入网址时，别省略URL的http://部分，否则程序将崩溃，因为Android不知道如何处理地址。在实际程序中，可以在用户省略了http://时自动将其添加上。

第2行用于新建一个`Intent`实例，将操作指定为`ACTION_VIEW`，并将要查看的对象指定为前面创建的`Uri`实例。最后，第3行调用方法`startActivity()`请求执行该操作。

活动Browser启动后，将创建自己的视图（如图10-2所示），并允许你的应用进入后台。用户按下“后退”按钮时，浏览器窗口将消失，而你的应用将恢复运行。

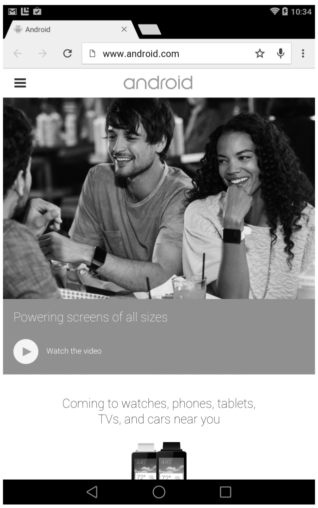


图　10-2

如果要同时显示你的应用的用户界面和网页，该如何处理呢？在Android中，可使用WebView类来实现这个目标。

10.2　使用 WebView 来浏览网页

在台式机中，Web浏览器是庞大而复杂的程序，它包含各种功能（如书签、插件、Flash动画、选项卡、滚动条、打印等），需要消耗大量的内存。

我在开发Eclipse项目时，有人建议将一些文本视图替换为嵌入式Web浏览器。我认为他们简直疯了。理由是，将文本视图进行改进，使其支持斜体、表格等缺失的功能不是更合理吗？

事实证明他们并没有疯，原因如下。

- 删掉除基本渲染引擎外的其他一切后，Web浏览器更为精炼。
- 如果改进文本视图，不断添加浏览器引擎提供的功能，最终得到的要么是过于复杂而臃肿的文本视图，要么是动力不足的浏览器。

Android提供了一个Chromium/WebKit浏览器引擎包装器——WebView，使用它可以获得浏览器的功能，而付出的开销却很小。

WebView的工作原理与其他Android视图很像，但它包含一些浏览器特有的方法。下面将创建前一个实例的嵌入版，以帮助你了解WebView的工作原理。我们将这个版本命名为BrowserView（而不是BrowserIntent），因为它使用的是嵌入式视图而不是意图。首先，使用如下设置新建一个项目。

- 应用名：BrowserView
- 公司域名：example.org
- 尺寸：Phone and Tablet
- 最低SDK：API 16: Android 4.1（Jelly Bean）
- 添加活动：Blank Activity
- 活动名：MainActivity
- 布局名：activity_main
- 标题：Browser View

BrowserView的布局文件与BrowserIntent的布局文件类似，但在最后面多了一个WebView。

browserView/src/main/res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">
```

```
    <LinearLayout
       android:orientation="horizontal"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content">
       <EditText
          android:id="@+id/url_field"
          android:layout_width="0dip"
          android:layout_height="wrap_content"
          android:layout_weight="1.0"
          android:lines="1"
          android:inputType="textUri"
          android:imeOptions="actionGo" />
       <Button
          android:id="@+id/go_button"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="@string/go_button" />
    </LinearLayout>
    <WebView
       android:id="@+id/web_view"
       android:layout_width="fill_parent"
       android:layout_height="0dip"
       android:layout_weight="1.0" />
</LinearLayout>
```

为了让所有界面元素都处于正确的位置，我们使用了两个`LinearLayout`控件。外面的`LinearLayout`控件将屏幕分为上下两部分。其中，上半部分包含文本区域和按钮，而下半部分包含`WebView`。内部的`LinearLayout`与前一个示例相同：将文本区域放在左边，并将按钮放在右边。

BrowserView的方法`onCreate()`与前一个示例相同，但还获取了指向新增视图的句柄。

browserView/src/main/java/org/example/browserview/MainActivity.java

```java
package org.example.browserview;
// ……
import android.webkit.WebView;
// ……

public class MainActivity extends Activity {
   private WebView webView;
   // ……
   @Override
   public void onCreate(Bundle savedInstanceState) {
      // ……
      webView = (WebView) findViewById(R.id.web_view);
      // ……
   }
}
```

然而，方法`openBrowser()`就不同了。

browserView/src/main/java/org/example/browserview/MainActivity.java

```
/** 打开浏览器并浏览到文本框中指定的URL */
private void openBrowser() {
   webView.getSettings().setJavaScriptEnabled(true);
   webView.loadUrl(urlText.getText().toString());
}
```

方法`loadUrl()`让浏览器引擎加载并显示指定地址的网页。这个方法会立即得到返回，虽然实际加载工作需要一段时间才能完成。

别忘了更新字符串资源。

browserView/src/main/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name">BrowserView</string>
   <string name="action_settings">Settings</string>
   <string name="go_button">Go</string>
</resources>
```

这个程序还有一个地方需要修改。在文件AndroidManifest.xml中，在标签`<application>`前面添加如下一行。

browserView/src/main/AndroidManifest.xml

```
<uses-permission android:name="android.permission.INTERNET" />
```

如果不这样做，Android将禁止应用访问Internet，进而显示错误Web page not available（网页不可用）。

> 小乔爱问：
>
> **BrowserIntent为何不需要`<uses-permission>`？**
>
> 在前一个示例（BrowserIntent）中，只是触发意图来请求另一个应用显示网页，该应用（浏览器）需要在其 AndroidManifest.xml 中请求 Internet 权限。

现在尝试运行这个程序，并输入以http://打头的有效网址。当你按回车键或Go按钮时，指定的网页将显示出来，如图10-3所示。

根据你输入的网址，可能需要在末尾添加字符/，这是因为我们没有处理重定向。

`WebView`包含数十个方法，你可以使用它们来控制显示的内容或收到有关状态变化的通知。完整的方法列表请参阅`WebView`的在线文档[①]。下面是你很可能需要使用的方法。

① http://d.android.com/reference/android/webkit/WebView.html

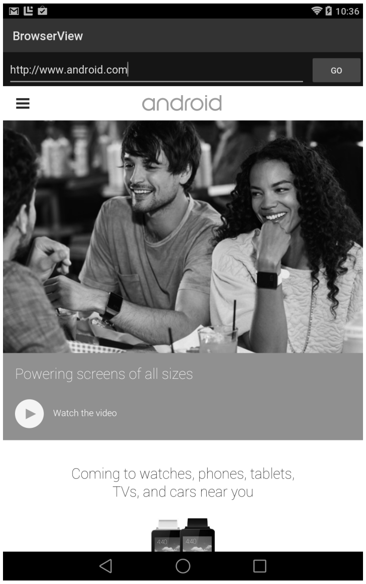


图　10-3

- addJavascriptInterface()：允许JavaScript访问Java对象（下一节将进行更详细的介绍）。
- createSnapshot()：创建当前网页的屏幕截图。
- getSettings()：返回一个用于控制设置的WebSettings对象。
- loadData()：将指定的字符串数据加载到浏览器中。
- loadDataWithBaseURL()：使用基本URL加载指定的数据。
- loadUrl()：加载指定URL处的网页。
- setDownloadListener()：为下载事件（如用户下载ZIP或APK文件）注册回调函数。
- setWebChromeClient()：为需要在WebView外面执行的事件（如更新标题或进度条，或者打开JavaScript对话框）注册回调函数。
- setWebViewClient()：允许应用在浏览器中设置钩子，以拦截资源加载、按键或申请授权等事件。
- stopLoading()：停止加载当前网页。

使用WebView控件时，可执行的最强大的任务是，在它和包含它的Android应用之间通信。下面就来更详细地介绍这项功能。

10.3　在 JavaScript 和 Java 之间交互

Android设备有很多卓越的功能，如存储本地数据、绘制图形、播放音乐、接打电话和定位。

如果能够在网页中访问这些功能，那该多好啊！使用嵌入式WebView控件，你就能够实现。

这里的关键是WebView类的方法addJavascriptInterface()。使用它可以扩展嵌入式浏览器中的文档对象模型（DOM，Document Object Model），还可以定义JavaScript代码能够访问的新对象。JavaScript代码调用这种对象的方法时，将实际调用Android程序中的方法。

你也可以在Android程序中调用JavaScript方法。为此，只需调用方法loadUrl()，并传入形式为javascript:code-to-execute的URL即可。这样，浏览器就可以在当前网页中执行指定的JavaScript表达式，而不是切换到新的网页。你可以调用方法、修改JavaScript变量、修改浏览器文档——应有尽有。

在JavaScript中调用Java的风险

每当允许网页访问本地资源或调用浏览器沙箱外的函数时，一定要仔细考虑这样做带来的安全后果。例如，你不会希望创建的方法让JavaScript根据任何路径名读取相应的数据，因为这可能会将机密数据暴露给知道该方法和文件名的恶意网站。

下面是一些需要牢记的要点。首先，不要依赖不明确的安全性，而应对哪些网页可使用你的方法进行限制，并对这些方法可执行的操作进行限制。其次，牢记如下安全黄金法则：**不要采用排除法，而应采用包含法**。换句话说，不要尝试去检查别人可能要求你做的所有坏事（如查询中的非法字符），这难免会挂一漏万；相反，应只允许别人做你知道安全的事情，其他的事情都不允许做（即检查查询只包含合法的字符）。

为演示WebView中的JavaScript和Android程序中的Java如何相互调用，下面来创建一个程序，它一半是HTML/JavaScript，一半是Android，如图10-4所示。在这个应用的窗口中，上半部分是一个WebView控件，下半部分是Android用户界面的TextView和Button。用户点击按钮或链接时，将发生两个环境之间的相互调用。

首先，使用如下设置新建一个程序。

- 应用名：LocalBrowser
- 公司域名：example.org
- 尺寸：Phone and Tablet
- 最低SDK：API 16: Android 4.1（Jelly Bean）
- 添加活动：Blank Activity
- 活动名：MainActivity
- 布局名：activity_main
- 标题：Local Browser

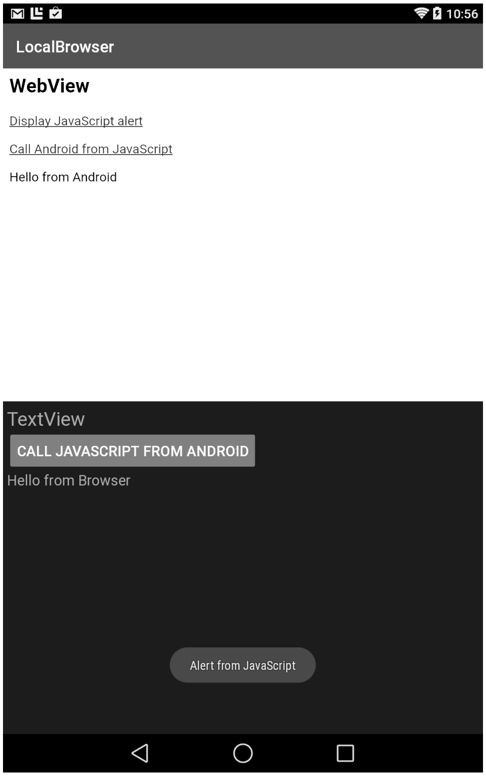


图　10-4

这个程序的用户界面分为两部分。第一部分是在Android布局文件res/layout/activity_main.xml中定义的。

localBrowser/src/main/res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">
   <WebView
      android:id="@+id/web_view"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1.0" />
   <LinearLayout
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1.0"
      android:padding="5sp">
```

```
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textSize="24sp"
            android:text="@string/textview" />
        <Button
            android:id="@+id/button"
            android:text="@string/call_javascript_from_android"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp" />
        <TextView
            android:id="@+id/text_view"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textSize="18sp" />
    </LinearLayout>
</LinearLayout>
```

第二部分是将加载到WebView中的文件index.html。这个文件存储在目录assets而不是res中，因为它并非编译型资源。程序安装时，目录assets的内容都将被原封不动地复制到本地存储区。这个目录用于存储HTML、图像和脚本的本地副本，浏览器无需连接到网络就能查看它们。

在Android Studio中，创建目录assets时稍微有点麻烦。首先必须将项目视图切换到Project模式，以便能够看到实际目录。为此，可单击窗口顶部的模式标签。切换到目录app/src/main，右击main，选择New➤Directory，输入目录名（assets），再单击OK按钮。接下来，返回到Android模式，右击assets并新建一个文件。

localBrowser/src/main/assets/index.html

```
<html>
<head>
<script language="JavaScript">
    function callJS(arg) {
        document.getElementById('replaceme').innerHTML = arg;
    }
</script>
</head>
<body>
<h2>WebView</h2>
<p>
<a href="#" onclick="window.alert('Alert from JavaScript')">
    Display JavaScript alert</a>
</p>
<p>
<a href="#" onclick="window.android.callAndroid('Hello from Browser')">
    Call Android from JavaScript</a>
</p>
<p id="replaceme">
</p>
</body>
</html>
```

在文件index.html中，第4行定义了Android程序将调用的函数`callJS()`。这个函数接受一个字符串参数，并将其插入到第19行的标签`replaceme`处。

在图10-4中，有两个HTML链接，它们是在始于第12行的代码中定义的。第一个链接调用标准函数`window.alert()`来打开一个窗口，并在其中显示一条简短的消息。第二个链接是在第16行定义的，它调用`window.android`对象的方法`callAndroid()`。如果在常规Web浏览器中加载该网页，对象`window.android`将是未定义的。然而，由于我们在Android应用中嵌入了一个浏览器，因此可以定义这个对象，让网页能够使用它。

接下来将注意力转向`MainActivity`类的Android代码。下面是基本轮廓，其中包含后面需要的所有`import`语句。

localBrowser/src/main/java/org/example/localbrowser/MainActivity.java

```
Line 1 package org.example.localbrowser;
     -
     - import android.app.Activity;
     - import android.os.Bundle;
     5 import android.os.Handler;
     - import android.util.Log;
     - import android.view.View;
     - import android.view.View.OnClickListener;
     - import android.webkit.JavascriptInterface;
    10 import android.webkit.JsResult;
     - import android.webkit.WebChromeClient;
     - import android.webkit.WebView;
     - import android.widget.Button;
     - import android.widget.TextView;
    15 import android.widget.Toast;
     -
     - public class MainActivity extends Activity {
     -    private static final String TAG = "LocalBrowser";
     -    private final Handler handler = new Handler();
    20    private WebView webView;
     -    private TextView textView;
     -    private Button button;
     -
     -    @Override
    25    public void onCreate(Bundle savedInstanceState) {
     -       super.onCreate(savedInstanceState);
     -       setContentView(R.layout.activity_main);
     -
     -       // 获取屏幕上的Android控件
    30       webView = (WebView) findViewById(R.id.web_view);
     -       textView = (TextView) findViewById(R.id.text_view);
     -       button = (Button) findViewById(R.id.button);
     -       // onCreate的其他代码……
     -    }
    35 }
```

注意到第19行初始化了一个`Handler`对象。JavaScript调用是在浏览器专用的特殊线程中进行

的，而Android用户界面调用只能在主（GUI）线程中进行。使用Handler类来实现这种切换。

线程是一个执行上下文，计算机代码在其中顺序执行（执行完一条语句后再执行另一条）。由于软件和硬件技术的进步，Android等现代操作系统可同时运行很多线程。其中一个是专用的前台（或GUI）线程，所有用户界面操作都是在这个线程中执行的。其他所有线程都是后台线程，用于执行长时间运行的操作（如网络I/O），以免影响用户体验。

要在JavaScript中调用Android Java代码，需要定义一个普通的Java对象（plain old Java object），其中包含一个或多个方法，如下所示。

localBrowser/src/main/java/org/example/localbrowser/MainActivity.java

```
/** 暴露给JavaScript的对象 */
private class AndroidBridge {
   @JavascriptInterface // 在Android 4.2+中必不可少
   public void callAndroid(final String arg) { // 必须为final
      handler.post(new Runnable() {
         public void run() {
            Log.d(TAG, "callAndroid(" + arg + ")");
            textView.setText(arg);
         }
      });
   }
}
```

JavaScript调用方法callAndroid()时，应用将新建一个Runnable对象，并使用Handler.post()将其发送到主线程的运行队列。主线程一有机会就会调用方法run()，而这个方法调用setText()来修改TextView控件显示的文本。现在，该在方法onCreate()中将一切组合起来了。首先，启用JavaScript（它默认被禁用），并向JavaScript注册前面定义的AndroidBridge。

localBrowser/src/main/java/org/example/localbrowser/MainActivity.java

```
// 在内嵌的浏览器中启用JavaScript
webView.getSettings().setJavaScriptEnabled(true);

// 在浏览器中向JavaScript暴露一个Java对象
webView.addJavascriptInterface(new AndroidBridge(),
      "android");
```


接下来，创建一个匿名的WebChromeClient对象，并使用方法setWebChromeClient()注册它。

localBrowser/src/main/java/org/example/localbrowser/MainActivity.java

```
// 创建一个函数，这个函数将在JavaScript试图打开提示窗口时被调用
webView.setWebChromeClient(new WebChromeClient() {
   @Override
   public boolean onJsAlert(final WebView view,
         final String url, final String message,
         JsResult result) {
      Log.d(TAG, "onJsAlert(" + view + ", " + url + ", "
```

```
            + message + ", " + result + ")");
        Toast.makeText(MainActivity.this, message, Toast.LENGTH_LONG).show();
        result.confirm();
        return true; // 已处理过
    }
});
```

在这里，术语chrome指的是浏览器窗口的所有点缀品（trimmings）。如果这是一个五脏俱全的浏览器客户端，将需要处理导航、书签、菜单等。在这里，我们只想改变浏览器（使用`window.alert()`）试图打开JavaScript提示框时发生的情况。在`onJsAlert()`中，使用Android类`Toast`创建一个消息窗口。这个窗口显示一小会儿之后就会消失。

配置好`WebView`后，便可使用`loadUrl()`来加载前述本地网页。

localBrowser/src/main/java/org/example/localbrowser/MainActivity.java

```
// 加载本地素材中的网页
webView.loadUrl("file:///android_asset/index.html");
```

对于Android的浏览器引擎来说，格式为file:///android_asset/filename（请注意，其中包含3个斜杠）的URL具有特殊含义。你可能猜到了，它们表示目录assets中的文件。在这里加载的是前面定义的文件index.html。

在示例LocalBrowser中，文件res/values/strings.xml的内容如下。

localBrowser/src/main/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name">LocalBrowser</string>
   <string name="action_settings">Settings</string>
   <string name="textview">TextView</string>
   <string name="call_javascript_from_android">
      Call JavaScript from Android
   </string>
</resources>
```

我们必须做的最后一项工作是，设置屏幕底部的按钮，使其执行JavaScript调用（在Java中调用JavaScript）。

localBrowser/src/main/java/org/example/localbrowser/MainActivity.java

```
// 用户点击按钮时，将在Android端调用这个函数
button.setOnClickListener(new OnClickListener() {
   public void onClick(View view) {
      Log.d(TAG, "onClick(" + view + ")");
      webView.loadUrl("javascript:callJS('Hello from Android')");
   }
});
```

为此，使用`setOnClickListener()`为按钮设置一个点击监听器。按钮被点击时，将调用

`onClick()`，而这个方法又会转而调用`WebView.loadUrl()`，并传入一个要在浏览器中执行的JavaScript表达式。这个表达式调用index.html中定义的函数`callJS()`。

现在运行这个程序。当你点击链接Display JavaScript alert时，将出现一个Android 消息窗口；当你点击链接Call Android from JavaScript时，字符串Hello from Browser将显示在一个Andriod `TextView`控件中；最后，当你点击按钮Call JavaScript from Android时，字符串Hello from Android将被发送给浏览器，而浏览器将把它插入HTML并显示在网页底部。

10.4 快速阅读指南

在本章中，你学习了两种显示Web内容的方式：启动意图来打开一个Web浏览器，以及在应用中嵌入一个Web浏览器。你还学习了如何使用`WebView`来显示本地内容，以及如何使用JavaScript在网页和Java之间通信。Apache Cordova[①]等跨平台工具包就使用了这种技术。

有时候，你不需要显示网页，而只想访问服务器提供的Web服务或其他数据。在下一章，你将学习如何实现这一目标。

如果你需要一个可带着去溜达的程序，可跳到第12章。

10

① http://cordova.apache.org

第 11 章 调用Web服务

近年来，越来越多的功能被移到了云端。Amazon Web Services提供了云计算和云存储，这是项数十亿美元的业务；Google Apps为数百万中小型企业处理办公功能；Microsoft正致力于将其无处不在的Office套件变成云服务。这些平台有一个共同之处，那就是都采用了遵循REST的Web服务接口。

REST（REpresentational State Transfer）的含义因人而异，但最具现实意义的定义是，这是一种在Internet上创建服务的技术，它能让用户通过TCP/IP（Transmission Control Protocol/Internet Protocol，传输控制协议/Internet协议）连接发送简单的HTTP（HyperText Transfer Protocol，超文本传输协议）来完成任务。简单地说，网上运行着服务器，用户可以使用标准协议通过标准通信端口连接到它，然后向它发送请求和命令并获得结果。

你每天都在使用的Web服务器（如google.com和microsoft.com）就是一种Web服务。客户端（浏览器）建立到端口80或443的连接，请求提供网页或其他资源，获得结果，再断开连接。整个万维网（World Wide Web）就是建立在这种简单架构基础之上的。

在本章中，你将学习如何在Android程序中建立到Web服务的网络连接，进而在应用中添加一系列新功能。

11.1 使用 Web 服务

Android提供了一整套Java标准联网API（如java.net.HttpURLConnection包），供你在程序中使用。最棘手的是进行异步调用，让程序的用户界面在任何时候都能快速响应。

如果在用户界面代码中间进行阻塞式网络调用，结果将如何呢？应用可能突然间不响应任何事件，如触摸或按钮点击。在用户看来，应用就像是停止了。显然，你必须避免这样的情况发生。

java.util.concurrent包非常适合用于完成这种工作。这个包最初是Doug Lea开发的一个独立库，后被加入到Java 5中。它支持并发编程，但抽象程度比Java类`Thread`高。`ExecutorService`类为你管理着一个或多个线程，你需要做的只是向它提交要执行的任务（`Runnable`或`Callable`实例）。`ExecutorService`将返回一个`Future`实例。它是一个引用，指向任务将返回的未来（当

前未知的）值。可以限制创建的线程数，还可以在必要的时候中断正在运行的任务。

为演示这些概念，下面来创建一个有趣的小程序，它调用Google Suggest API。你可能注意到了，在google.com或bing.com等搜索网站搜索时，输入关键字后，你将立即看到补全建议。例如，如果你输入字母and，可能会看到一些与Android相关的建议。这是使用Web服务实现的。

其工作原理如下：每当你在搜索框中输入字符时，浏览器或网页都会调用服务器，看看有哪些以你输入的字母打头的内容。因此，当你输入a时，将返回一些以a打头的短语。当你接着又输入n时，将返回一些以an打头的短语。依此类推。

在服务器内部，进行了一些非常聪明的处理。它知道你是谁、身处何方，以及最近搜索了什么，因此能够为你定制结果。请尝试做个这样的实验：与朋友一起将各自的计算机并排放置，并访问同一个搜索网站。然后，以每次一个字母的方式输入相同的短语，并比较你们得到的建议——建议很可能截然不同。按下回车键后，最终返回的结果也将不同[①]。

11.2 Suggest 示例

下面来创建一个程序，它调用Web服务Suggest并显示结果，就像搜索引擎或智能地址栏一样。要使用这个程序，只需输入一个短语即可。在你输入时，这个程序将使用Web服务Suggest获取补全建议，如图11-1所示。

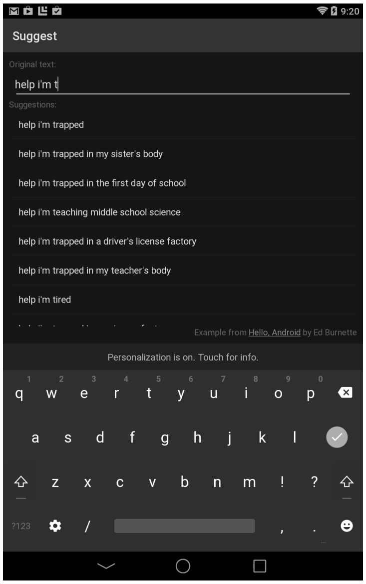


图 11-1

① 有关这种现象的更详细信息，请参阅http://en.wikipedia.org/wiki/Filter_bubble。

为创建这个应用，首先使用如下设置新建一个项目。

- 应用名：Suggest
- 公司域名：example.org
- 尺寸：Phone and Tablet
- 最低SDK：API 16: Android 4.1（Jelly Bean）
- 添加活动：Blank Activity
- 活动名：MainActivity
- 布局名：activity_main
- 标题：Suggest

由于这个示例要访问Internet以调用Web服务，因此需要让Android授予我们这种权限。

为此，在文件AndroidManifest.xml中，在XML标签`<application>`前面添加如下一行。

suggest/src/main/AndroidManifest.xml

```
<uses-permission android:name="android.permission.INTERNET" />
```

主活动的布局非常简单，为一个包含多行的垂直`LinearLayout`。

suggest/src/main/res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:padding="10dip" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/original_label" />

    <EditText
        android:id="@+id/original_text"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/original_hint"
        android:inputType="textNoSuggestions"
        android:padding="10dip"
        android:textSize="18sp" />

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/result_label" />

    <ListView
        android:id="@+id/result_list"
```

```
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1" />

    <TextView
        android:id="@+id/eband_text"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="bottom|right"
        android:text="@string/eband" />

</LinearLayout>
```

在这个示例中，包含5行。第1行是一个标签，让用户在第2行（一个文本框）中输入要搜索的内容。第3行是另一个标签，后面是建议列表。我要将这个示例发布到Google Play Store，因此最后一行是为本书做的一个小广告。

下面来修改MainActivity类，其基本轮廓如下。

suggest/src/main/java/org/example/suggest/MainActivity.java

```
package org.example.suggest;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;
import java.util.concurrent.RejectedExecutionException;

import android.app.Activity;
import android.app.SearchManager;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.text.Editable;
import android.text.TextWatcher;
import android.text.method.LinkMovementMethod;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;

public class MainActivity extends Activity {
   private EditText origText;
   private ListView suggList;
   private TextView ebandText;

   private Handler guiThread;
   private ExecutorService suggThread;
```

```
-    private Runnable updateTask;
-    private Future<?> suggPending;
35   private List<String> items;
-    private ArrayAdapter<String> adapter;
-
-    @Override
-    public void onCreate(Bundle savedInstanceState) {
40      super.onCreate(savedInstanceState);
-
-       setContentView(R.layout.activity_main);
-       initThreading();
-       findViews();
45      setListeners();
-       setAdapters();
-
-    }
- }
```

声明几个变量后，从第39行开始定义了方法`onCreate()`，它负责初始化线程和用户界面。请别担心，后面将编写它调用的所有方法。

第44行调用的方法`findViews()`用来获取句柄，它们分别指向布局文件定义的各个用户界面元素。

suggest/src/main/java/org/example/suggest/MainActivity.java

```
private void findViews() {
   origText = (EditText) findViewById(R.id.original_text);
   suggList = (ListView) findViewById(R.id.result_list);
   ebandText = (TextView) findViewById(R.id.eband_text);
}
```

在方法`onCreate()`中，第46行调用的方法`setAdapters()`负责为列表视图定义数据源。

suggest/src/main/java/org/example/suggest/MainActivity.java

```
/** 设置列表视图的适配器 */
private void setAdapters() {
   items = new ArrayList<String>();
   adapter = new ArrayAdapter<String>(this,
         android.R.layout.simple_list_item_1, items);
   suggList.setAdapter(adapter);
}
```

在Android中，`Adapter`是一个将数据源（这里是建议列表）绑定到用户界面控件（这里是一个`ListView`）的类。对于每个列表项，都使用Android提供的标准布局。

接下来，在方法`setListeners()`中（在方法`onCreate()`中，这是在第45行调用的）设置用户界面处理程序。

suggest/src/main/java/org/example/suggest/MainActivity.java

```
private void setListeners() {
   // 定义文本变化监听器
   TextWatcher textWatcher = new TextWatcher() {
      public void beforeTextChanged(CharSequence s, int start,
            int count, int after) {
         /* 什么都不做 */
      }
      public void onTextChanged(CharSequence s, int start,
            int before, int count) {
         queueUpdate(1000 /* 毫秒数 */);
      }
      public void afterTextChanged(Editable s) {
         /* 什么都不做 */
      }
   };

   // 给搜索框指定监听器
   origText.addTextChangedListener(textWatcher);

   // 定义列表项点击监听器
   OnItemClickListener clickListener = new OnItemClickListener() {
      @Override
      public void onItemClick(AdapterView<?> parent, View view,
            int position, long id) {
         String query = (String) parent.getItemAtPosition(position);
         doSearch(query);
      }
   };

   // 给建议列表指定监听器
   suggList.setOnItemClickListener(clickListener);

   // 将网站链接设置成可点击的
   ebandText.setMovementMethod(LinkMovementMethod.getInstance());
}

private void doSearch(String query) {
   Intent intent = new Intent(Intent.ACTION_WEB_SEARCH);
   intent.putExtra(SearchManager.QUERY, query);
   startActivity(intent);
}
```

这里定义了两个监听器：一个在输入文本框的内容发生变化时被调用，另一个在用户点击建议时被调用。方法`queueUpdate()`使用`Handler`将一个延迟更新请求加入到主线程的待办事项列表中，这里随意地将文本变化延迟设置成了1000毫秒。

11

在方法`setListeners()`的最后，对广告视图调用了方法`setMovementMethod()`，将广告文本转换成了超链接。这样，在用户点击该链接时，将打开一个浏览器窗口，并在其中显示指定地址处的网页。

11.3　穿针引线

更新请求是在方法`initThreading()`中定义的。

suggest/src/main/java/org/example/suggest/MainActivity.java

```
Line 1 private void initThreading() {
     -    guiThread = new Handler();
     -    suggThread = Executors.newSingleThreadExecutor();
     -
     5    // 这个任务要求获取建议并更新屏幕
     -    updateTask = new Runnable() {
     -       public void run() {
     -          // 获取搜索文本
     -          String original = origText.getText().toString().trim();
    10
     -          // 撤销以前的建议（如果有的话）
     -          if (suggPending != null)
     -             suggPending.cancel(true);
     -
    15          // 确认输入了搜索文本
     -          if (original.length() != 0) {
     -             // 让用户知道程序正在执行操作
     -             setText(R.string.working);
     -
    20             // 开始获取建议，但不等待结果
     -             try {
     -                SuggestTask suggestTask = new SuggestTask(
     -                      MainActivity.this, // 指向当前活动的引用
     -                      original // 搜索文本
    25                );
     -                suggPending = suggThread.submit(suggestTask);
     -             } catch (RejectedExecutionException e) {
     -                // 无法开始新任务
     -                setText(R.string.error);
    30             }
     -          }
     -       }
     -    };
     - }
```

我们有两个线程：用于用户界面的主Android线程和用于执行实际建议作业的建议线程。其中，第一个线程由一个Android `Handler`表示，第二个线程由Java `ExecutorService`表示。

第6行定义了将由方法`queueUpdate()`调度的更新任务。这个任务在运行时，首先会获取当前的搜索文本，然后准备好将建议作业发送给建议线程。撤销还未完成的建议作业（第13行），确认用户输入了搜索文本（第16行），并在建议列表中显示字符串Working（第18行）。后面将把该文本替换为实际建议。

最后，第22行将创建一个`SuggestTask`实例。我们给它提供了指向主活动的引用，让它能够修改建议列表。我们还提供了一个包含搜索文本的字符串。第26行将这个新任务提交给建议线程。

这将返回一个引用，它指向最终以Future的方式返回的值。在这个示例中，实际上并不需要返回值，因为SuggestTask直接修改了GUI，但前面的第13行使用了这个Future引用来撤销未完成的建议获取作业。

小乔爱问：

有必要使用延迟和线程技术吗？

这样做的目的之一是避免过多地调用外部Web服务。想象一下用户在输入单词scissors时发生的情况吧。在这个程序看来，这个单词是以每次一个字符的方式输入的。首先是s，然后依次为c、i等，还可能有退格，因为用户可能忘记了这个单词是如何拼写的。你希望用户每输入一个字符就发起一次Web服务请求吗？当然不希望。这样做除了会给服务器增加不必要的负担外，还会浪费电量。每次请求时，设备的无线设备都得发送和接收多个数据包，这将耗费一定的电量。你会希望等到用户输入完毕后再发送请求，但如何判断用户输入完毕了呢？

这里使用的算法是，用户每输入一个字符，就发起一个延迟的请求。如果延迟1秒后用户还没有输入另一个字符，请求将发送出去；否则就将前一个请求从请求队列中删除。如果请求已发出，但还未完成，就尝试中断它。好消息是，经过我的示范后，你可以将这种技巧用于自己的异步程序中。

11.4　细枝末节

在Suggest示例中，还包含其他一些工具函数，如下所示。

suggest/src/main/java/org/example/suggest/MainActivity.java

```java
/** 请求短暂延迟后进行更新 */
private void queueUpdate(long delayMillis) {
   // 撤销以前的更新（如果它还没有开始）
   guiThread.removeCallbacks(updateTask);
   // 如果几毫秒后什么都没有发生，就开始更新
   guiThread.postDelayed(updateTask, delayMillis);
}

/** 修改屏幕上的建议列表（从另一个线程中调用） */
public void setSuggestions(List<String> suggestions) {
   guiSetList(suggList, suggestions);
}

/** 对GUI的所有修改都必须在GUI线程中进行 */
private void guiSetList(final ListView view,
      final List<String> list) {
```

```
      guiThread.post(new Runnable() {
         public void run() {
            setList(list);
         }

      });
   }

   /** 显示一条消息 */
   private void setText(int id) {
      adapter.clear();
      adapter.add(getResources().getString(id));
   }

   /** 显示建议列表 */
   private void setList(List<String> list) {
      adapter.clear();
      adapter.addAll(list);
   }
```

queueUpdate()负责将一个更新请求加入主线程的请求队列中，但要求等待一段时间后再运行它。如果队列中已经有请求存在，就将其删除。

Web服务返回结果后，SuggestTask将使用方法setSuggestions()来更新用户界面。这个方法调用了私有方法guiSetList()，后者又会调用方法Handler.post()请求主GUI线程更新列表视图。这一步必不可少，因为你不能在非用户界面线程中调用用户界面函数，而guiSetList()是在建议线程中被调用的。

最后，由setText()和setList()更新适配器，使其包含单个字符串或一个字符串列表。

下面是Suggest示例中的文件res/values/strings.xml，它定义了该程序使用的字符串资源。

suggest/src/main/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Suggest</string>
    <string name="action_settings">Settings</string>
    <string name="original_hint">Enter partial text</string>
    <string name="original_label">Original text:</string>
    <string name="result_label">Suggestions:</string>
    <string name="working">Working...</string>
    <string name="error">(Web service error)</string>
    <string name="interrupted">(Web service interrupted)</string>
    <string name="no_results">(No suggestions)</string>
    <string name="eband">Example from
       <a href="http://pragprog.com/book/eband4">Hello, Android</a>
       by Ed Burnette</string>

</resources>
```

到现在还没有介绍的只有SuggestTask类了，它负责在后台调用Web服务。

11.5 建议获取任务

最后，SuggestTask的定义如下。

suggest/src/main/java/org/example/suggest/SuggestTask.java

```
package org.example.suggest;

import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.util.LinkedList;
import java.util.List;

import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;

import android.util.Log;
import android.util.Xml;

public class SuggestTask implements Runnable {
   private static final String TAG = "SuggestTask";
   private final MainActivity suggest;
   private final String original;

   SuggestTask(MainActivity context, String original) {
      this.suggest = context;
      this.original = original;
   }

   public void run() {
      // 获取针对搜索文本的建议
      List<String> suggestions = doSuggest(original);
      suggest.setSuggestions(suggestions);
   }

   /**
    * 调用Google Suggest API，根据搜索文本创建一个建议列表
    *
    * 注意：这个API可能未得到支持。如果它不管用，请尝试使用Yahoo提供的API
    *
    * http://ff.search.yahoo.com/gossip?output=xml&command=WORD 或
    * http://ff.search.yahoo.com/gossip?output=fxjson&command=WORD
    */
   private List<String> doSuggest(String original) {
      List<String> messages = new LinkedList<String>();
      String error = null;
      HttpURLConnection con = null;
      Log.d(TAG, "doSuggest(" + original + ")");

      try {
         // 检查任务是否被中断
```

11

```
    if (Thread.interrupted())
       throw new InterruptedException();

    // 创建针对Google API的RESTful查询
    String q = URLEncoder.encode(original, "UTF-8");
    URL url = new URL(
          "http://google.com/complete/search?output=toolbar&q="
                + q);
    con = (HttpURLConnection) url.openConnection();
    con.setReadTimeout(10000 /* 毫秒数 */);
    con.setConnectTimeout(15000 /* 毫秒数 */);
    con.setRequestMethod("GET");
    con.addRequestProperty("Referer",
          "http://www.pragprog.com/book/eband4");
    con.setDoInput(true);

    // 启动查询
    con.connect();

    // 检查任务是否被中断
    if (Thread.interrupted())
       throw new InterruptedException();

    // 提取查询结果
    XmlPullParser parser = Xml.newPullParser();
    parser.setInput(con.getInputStream(), null);
    int eventType = parser.getEventType();
    while (eventType != XmlPullParser.END_DOCUMENT) {
       String name = parser.getName();
       if (eventType == XmlPullParser.START_TAG
             && name.equalsIgnoreCase("suggestion")) {
          for (int i = 0; i < parser.getAttributeCount(); i++) {
             if (parser.getAttributeName(i).equalsIgnoreCase(
                   "data")) {
                messages.add(parser.getAttributeValue(i));
             }
          }
       }
       eventType = parser.next();
    }

    // 检查任务是否被中断
    if (Thread.interrupted())
       throw new InterruptedException();

 } catch (IOException e) {
    Log.e(TAG, "IOException", e);
    error = suggest.getResources().getString(R.string.error)
          + " " + e.toString();
 } catch (XmlPullParserException e) {
    Log.e(TAG, "XmlPullParserException", e);
    error = suggest.getResources().getString(R.string.error)
          + " " + e.toString();
 } catch (InterruptedException e) {
```

```
            Log.d(TAG, "InterruptedException", e);
            error = suggest.getResources().getString(
                  R.string.interrupted);
         } finally {
            if (con != null) {
               con.disconnect();
            }
         }

         // 如果发生了错误，就返回错误本身
         if (error != null) {
            messages.clear();
            messages.add(error);
         }

         // 指出没有查询结果
         if (messages.size() == 0) {
            messages.add(suggest.getResources().getString(
                  R.string.no_results));
         }

         // 处理完毕
         Log.d(TAG, "   -> returned " + messages);
         return messages;
      }
   }
```

这是一个很不错的示例，它使用`HttpURLConnection`调用了一个RESTful Web服务，来对JSON（JavaScript Object Notation）格式的结果进行分析，并处理了各种网络错误和中断请求。这里不打算详细阐述它，因为除了几条调试消息外，其他代码都不是Android特有的。

至此，Suggest示例就编写完成了。请尝试运行一下，看看Google会认为你要搜索什么有趣的东西。只要进行简单的搜索，你就可以在网上找到更多有趣的搜索建议。

11.6 快速阅读指南

本章介绍了如何使用异步Web服务。如果你要使用`ExecutorService`等类进行大量的并发编程，建议阅读Brian Goetz编著的《JAVA并发编程实战》（*Java Concurrency in Practice*）[Goe06]。

创建Web服务不在本书的讨论范围之内，但最简单的方法之一是使用Google App Engine（GAE）[1]，它能够让你在云端托管使用各种语言（如Java、Python和PHP）编写的后端服务。

后面的章节将带领你探索如何使用触摸事件和Google服务提供另一种交互。如果你急于更深入地了解数据源和数据绑定，可直接跳到第13章。其中介绍了另一种减轻GUI线程负担的方式——使用`Loader`类。

11

① http://developers.google.com/appengine/

第 12 章

使用Google Play服务

12

Google Play服务是Android框架的一个插件，可以在任何使用Google Play Store的Android设备中找到。

Google Play服务最初默默无闻，只是用来访问Google授权和社交网络Google+的一种方式，然而现在却已经拥有了大量的功能，其中很多最初都包含在Android框架中。这些功能包括：

- 位置服务
- 游戏服务
- 应用内购买
- 广告
- 移动分析（Mobile analytics）
- 地图
- 推送消息
- 云存储
- 其他

在应用中包含、配置和调用所有Google Play服务的方式都相似。因此只要学习一种服务，学会使用所有服务的任务便完成了一半。

本章将以位置服务API为例简要地介绍Google Play服务。阅读完本章后，你将了解如何在应用中包含Google Play服务，如何检测它是否可用，如何调用它，以及如何处理发生的各种错误。

12.1 工作原理

在你的应用中包含一个小型客户端库，它负责与Google Play服务APK（应用包）通信。当你执行调用时，小型客户端库会向服务发送一条消息，而服务将以你的名义执行指定的操作。

每个应用都与同一个服务APK通信（如图12-1所示），该APK会通过Google Play Store频繁地进行更新。这种设计能够让Google将修复补丁和新功能直接推送到你的手机或平板电脑，而无需等待Android系统更新和运营商批准。这还可以节省空间，因为不要求每个应用重复包含相同的代码。

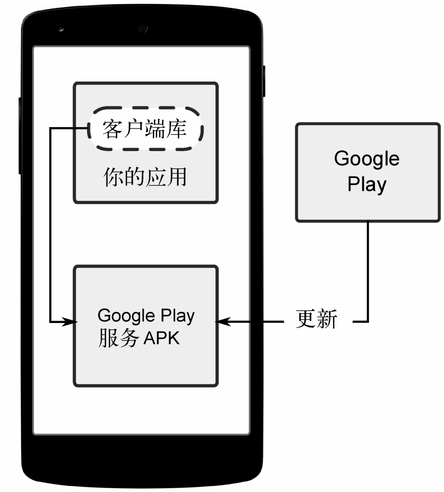


图 12-1

在Google Play服务提供的功能中，最常用的一个是位置服务。下面深入介绍其用途和用法。

12.2 使用位置服务

几乎每部手机都装备了GPS和加速计芯片等传感器，这打开了通向位置和环境识别移动计算的大门。宽带移动网络日益普及，计算和存储能力呈几何级数增长，这些都给人类彼此交互和人机交互的方式带来了革命性改变。

Android框架提供了大量可用于在应用中集成传感器数据的API，包括从GPS、基站和Wi-Fi获取位置信息的低级API。解读所有这些数据并不容易，但位置服务API可助你一臂之力。位置服务提供了一个强大的高级框架，所有应用都可使用它来获得可靠的位置数据，同时减少消耗的电量。

位置服务最初是一个自定义库，仅供Google Maps和其他专用应用使用。通过将其加入到Google Play服务中，Google能够让你在自己的应用中使用该技术。当前，Google建议编写新代码时都使用位置服务。

为演示位置服务，我们将创建一个应用，它显示随时间推移而不断变化的当前位置。图12-2是这个项目的屏幕截图。

这个应用首先显示设备的初始位置，然后频繁地更新屏幕，不断显示最新的位置。如果你坐着不动，就看不出它的优势。这是一个适合带着去散步的应用。

图12-2中的各个数字都是什么意思呢？第一栏的数字是相邻两行的打印间隔，接下来是以字符串方式打印的位置。这些字符串的格式是使用方法`Location.toString()`设置的。

12

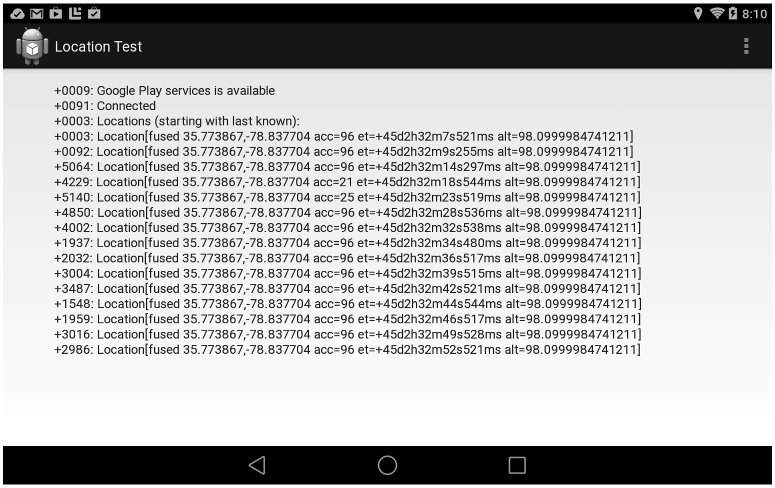


图　12-2

在Java编程中，一种常见的做法是，为每个主要类都定义方法`toString()`。这样可以在调试期间轻松地打印它们。`Location`类的实例以如下格式进行打印。

```
Location[<provider-type> <longitude>, <latitude>, acc=<accuracy>]
```

其中各项内容的含义如下。

- `provider-type`：计算位置的硬件或软件提供器。Fused提供器是一款智能软件，它结合使用各种硬件提供器（Wi-Fi、基站等）提供的位置来获得最准确的位置，同时最大限度地减少耗电量。
- `longitude`和`latitude`：在地球上的当前位置。
- `accuracy`：系统测量位置的准确度。例如，基站定位很不准确，如果系统使用的是这种方式，相应的数字将很大。

你还可能看到其他信息（如果这些信息可用），如过去了多长时间、海拔、航向和速度。

12.2.1　起步

说得够多了，现在开始编码吧！使用如下设置新建一个应用。

- 应用名：Location Test
- 公司域名：example.org
- 尺寸：Phone and Tablet
- 最低SDK：API 16: Android 4.1（Jelly Bean）
- 添加活动：Blank Activity
- 活动名：MainActivity

- 布局名：activity_main
- 标题：LocationTest

设置使用Google Play服务的项目时，相比于普通Andriod应用，这里需要执行两三个额外的步骤。首先，需要修改编译脚本，在其中添加对Google Play服务客户端库的编译阶段依赖。为此，打开app模块（而不是Project:LocationTest模块）的build.gradle文件，在`dependencies`下新增一条使用最新版`play-services`的编译规则，如下所示。

locationTest/build.gradle

```
// ……
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:22+'
    compile 'com.google.android.gms:play-services:7.+'
}
```

在窗口顶部将出现一条警告消息，指出最后一次项目同步后Gradle文件发生了变化（Gradle files have changed since last project sync）。单击Sync Now链接重新编译项目。

第二个额外的步骤是，需要在清单文件中添加一项信息。为此，打开文件AndroidManifest.xml，并在`application`标签下添加如下标签。

locationTest/src/main/AndroidManifest.xml

```
<meta-data
      android:name="com.google.android.gms.version"
      android:value="@integer/google_play_services_version" />
```

如果要使用ProGuard（一个应用优化和代码混淆工具），还需执行一个额外步骤。本书的示例都没有使用它，但如果你的应用使用到了，还必须编辑文件proguard-project.txt，以防ProGuard剥离必不可少的类。有关这方面的详情，请参阅在线文档[①]。

设置好项目后，下面来处理用户界面。

12.2.2　创建用户界面

主活动包含一个可滚动的文本视图，它覆盖了整个屏幕。日志输出将被添加到末尾，填满后文本将向上滚动。请编辑布局文件activity_main.xml，在其中定义一个id为`scroller`的ScrollView，它包含一个id为`output`的TextView。

locationTest/src/main/res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
   xmlns:android="http://schemas.android.com/apk/res/android"
```


① http://d.android.com/google/play-services/setup.html

```
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/scroller"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/output"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:textSize="16sp" />
</ScrollView>
```

将文本大小设置为16sp，让文本更容易识别。

接下来，打开文件MainActivity，并创建如下轮廓。

locationTest/src/main/java/org/example/locationtest/MainActivity.java

```
package org.example.locationtest;

import android.app.Activity;
import android.app.Dialog;
import android.content.Intent;
import android.content.IntentSender;
import android.location.Location;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ScrollView;
import android.widget.TextView;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GooglePlayServicesUtil;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;

public class MainActivity extends Activity implements
      GoogleApiClient.ConnectionCallbacks,
      GoogleApiClient.OnConnectionFailedListener,
      LocationListener {
   private static final long UPDATE_INTERVAL = 5000;
   private static final long FASTEST_INTERVAL = 1000;
   private static final int CONNECTION_FAILURE_RESOLUTION_REQUEST = 9000;

   private TextView mOutput;
   private ScrollView mScroller;
```

```
-    private GoogleApiClient mGoogleApiClient;
-    private long mLastTime;
-    // ……
35 }
```

从第3行开始，是这个类需要的所有`import`语句。如果是手动输入这个示例的代码，可以不输入这些`import`语句，而让Android Studio自动添加它们，即在每个未定义引用处单击鼠标，再按Alt和回车键。

`MainActivity`类始于第22行。它继承了`Activity`类，并实现了两个接口：`ConnectionCallbacks`和`OnConnectionFailedListener`。实现接口是为了告诉编译器，我们将添加这些接口定义的必要方法。添加这些方法前，编辑器窗口中始终有错误标记。

在`MainActivity`中，我们声明了一些后面需要的常量和变量。

有了基本轮廓后，来编写缺失的方法。第一个是活动启动前调用的方法`onCreate()`。

locationTest/src/main/java/org/example/locationtest/MainActivity.java

```
Line 1 @Override
-  public void onCreate(Bundle savedInstanceState) {
-     super.onCreate(savedInstanceState);
-     setContentView(R.layout.activity_main);
5
-     // 定义前述API客户端
-     mGoogleApiClient = new GoogleApiClient.Builder(this)
-           .addConnectionCallbacks(this)
-           .addOnConnectionFailedListener(this)
10         .addApi(LocationServices.API)
-           .build();
-
-     // 获取指向视图的引用
-     mOutput = (TextView) findViewById(R.id.output);
15    mScroller = (ScrollView) findViewById(R.id.scroller);
-
-     // 获取当前时间，以便获悉更新之后的时间
-     mLastTime = System.currentTimeMillis();
-  }
```

首先使用根据布局XML生成的视图填充活动，如第4行所示。第7行用于新建一个`GoogleApiClient`（Google Play服务中的一个类）实例，后面将连接到这个实例来获取位置。

第14~15行调用`findViewById()`获取指向布局中两个视图的句柄。最后，第18行获取当前的系统时间，以便计算打印每行日志时过去了多少时间。

下面的两个方法是Blank Activity模板添加的。

locationTest/src/main/java/org/example/locationtest/MainActivity.java

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   // 生成菜单，并将菜单项加入到操作栏中
```

```java
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // 处理操作栏项点击事件。只要你在AndroidManifest.xml中指定了父活动，
    // 操作栏就会自动处理主屏幕/向上按钮点击事件
    int id = item.getItemId();
    if (id == R.id.action_settings) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

这些方法实现了一个Settings（设置）菜单，可以用来在程序中添加选项。例如，可以添加控制位置更新频率或位置精度的选项。我将这项任务作为练习留给你去完成。就现在而言，保留这些方法不变即可。

12.2.3　连接到位置提供器

要在用户启动该应用或从其他应用切换到该应用时连接到位置提供器，并在用户从该应用切换到其他应用时断开连接。在方法`onResume()`和`onPause()`中分别执行这些操作最合适。

locationTest/src/main/java/org/example/locationtest/MainActivity.java

```java
@Override
protected void onResume() {
    super.onResume();
    // 如果Google Play服务可用，就连接到客户端
    if (servicesAvailable()) {
        mGoogleApiClient.connect();
    }
}

@Override
protected void onPause() {
    if (mGoogleApiClient.isConnected()) {
        mGoogleApiClient.disconnect();
    }
    super.onPause();
}
```

在`onResume()`中检查Google Play服务是否可用。如果可用，就对前面在方法`onCreate()`中创建的Google API客户端调用方法`connect()`。在方法`onPause()`中检查是否连接到了Google API客户端。如果连接到了Google API客户端，就断开连接。

调用方法`connect()`有一个重要的副作用：成功建立连接后，位置服务将调用你的类的方法`onConnected()`。下面演示如何定义这个方法。

locationTest/src/main/java/org/example/locationtest/MainActivity.java

```java
/**
 * 成功连接到Google API客户端后，位置服务将调用这个方法。
 * 在这个方法中，你可以请求当前位置或启动定期更新
 */
@Override
public void onConnected(Bundle dataBundle) {
   // 显示连接状态
   log("Connected");

   // 获取当前位置
   Location location = LocationServices.FusedLocationApi.getLastLocation(
         mGoogleApiClient);
   log("Locations (starting with last known):");
   if (location != null) {
      dumpLocation(location);
   }
   // ……
}
```

方法log()将在输出视图中添加一行内容，并向上滚动输出视图。方法dumpLocation()用于将位置转换为字符串并将其加入到输出视图中。

locationTest/src/main/java/org/example/locationtest/MainActivity.java

```java
/** 将一个字符串写入到输出窗口中 */
private void log(String string) {
   long newTime = System.currentTimeMillis();
   mOutput.append(String.format("+%04d: %s\n",
         newTime - mLastTime, string));
   mLastTime = newTime;

   // 将文本视图滚动到末尾的小窍门
   mScroller.post(new Runnable() {
      @Override
      public void run() {
         mScroller.fullScroll(View.FOCUS_DOWN);
      }
   });
}

/** 描述给定的位置，它可能为null */
private void dumpLocation(Location location) {
   if (location == null)
      log("Location[unknown]");
   else
      log(location.toString());
}
```


注意到，打印每条日志消息前都会获取当前时间，并将其与前一次的时间相减，在行首打印时间差。

12.2.4　获取更新

当前，这个程序只在启动时显示用户的位置。我们希望它频繁地打印位置（这样在用户移动时，日志将越来越长），并显示用户经过的所有位置。为此，只需进行3处简单修改即可。

首先，在方法`onConnected()`的末尾调用一个方法，使用API客户端来请求更新。

代码如下。

locationTest/src/main/java/org/example/locationtest/MainActivity.java

```
public void onConnected(Bundle dataBundle) {
   // ……
   // 请求高精度的更新，更新间隔为1~5秒
   LocationRequest locationRequest = LocationRequest.create();
   locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
   locationRequest.setInterval(UPDATE_INTERVAL);
   locationRequest.setFastestInterval(FASTEST_INTERVAL);
   LocationServices.FusedLocationApi.requestLocationUpdates(
         mGoogleApiClient, locationRequest, this);
}
```

在这个示例中，我们设置了精度优先于电源消耗的优先级，并要求至少5秒更新一次。如果另一个应用也使用位置服务，但获取位置的间隔短于5秒，我们的应用获取更新的频率将更高。然而，我们还设置了最高频率，来告诉系统，我们不希望每秒获取的更新超过一个。

这些设置只是提示。系统会尽可能满足你的愿望，但为了节省电量，它可能会自作主张。这意味着你应做好这样的准备，即更新频率可能比你要求的高或低。运行这个应用时，如果你查看每行的打印时间，就能清楚地看到这一点。

有更新可用时，将调用方法`onLocationChanged()`。

locationTest/src/main/java/org/example/locationtest/MainActivity.java

```
@Override
public void onLocationChanged(Location location) {
   dumpLocation(location);
}
```

这个方法只是用来将位置打印到日志视图。

12.2.5　处理错误

没有Google Play Store的设备（其中最著名的是Amazon Kindle Fire）没有Google Play服务。在方法`onResume()`中，调用了方法`servicesAvailable()`，用来检查是否安装了Google Play服务，以及该服务是否是最新的。这个方法的定义如下。

locationTest/src/main/java/org/example/locationtest/MainActivity.java

```
/** 检查Google Play服务是否可用 */
private boolean servicesAvailable() {
   int resultCode = GooglePlayServicesUtil.
         isGooglePlayServicesAvailable(this);
   if (ConnectionResult.SUCCESS == resultCode) {
      log("Google Play services is available");
      return true;
   } else {
      log("Google Play services is not available");
      showErrorDialog(resultCode);
      return false;
   }
}
```

如果Google Play服务不可用，就调用方法`showErrorDialog()`。

locationTest/src/main/java/org/example/locationtest/MainActivity.java

```
/** 显示一条Google Play服务错误消息 */
private void showErrorDialog(int resultCode) {
   // 从Google Play服务获取错误对话框
   Dialog errorDialog = GooglePlayServicesUtil.getErrorDialog(
         resultCode, this, CONNECTION_FAILURE_RESOLUTION_REQUEST);

   if (errorDialog != null) {
      // 显示错误
      errorDialog.show();
   }
}
```

Google Play服务客户端库提供了一个创建错误对话框以将错误显示给用户的函数。

连接到位置服务时也可能发生错误。建立连接可能会需要一段时间。如果连接成功，将调用活动的方法`onConnected()`；如果连接失败，将调用方法`onConnectionFailed()`。

locationTest/src/main/java/org/example/locationtest/MainActivity.java

```
/**
 * 如果在试图连接到位置客户端时以失败告终，位置服务将调用这个方法
 */
@Override
public void onConnectionFailed(ConnectionResult connectionResult) {
   // 能够消除错误（如通过安装新版本）吗
   log("Connection failed");
   if (connectionResult.hasResolution()) {
      try {
         // 启动一个尝试消除错误的活动
         log("Trying to resolve the error...");
         connectionResult.startResolutionForResult(
               this, CONNECTION_FAILURE_RESOLUTION_REQUEST);
   } catch (IntentSender.SendIntentException e) {
      log("Exception during resolution: " + e.toString());
   }
```

12

```
} else {
   // 没有找到错误解决方案
   showErrorDialog(connectionResult.getErrorCode());
   }
}

/**
 * 客户端暂时处于断开状态时将调用这个方法
 */
@Override
public void onConnectionSuspended(int cause) {
   log("Connection suspended");
}
```

假设有Google Play服务可用，但不是（AndroidManifest.xml中的元数据）指定的版本。在这种情况下，可尝试消除错误后再重试：显示一个对话框，其中包含一个让用户前往Google Play Store下载正确版本的选项。用户下载正确的版本后，将调用方法`onActivityResult()`。

locationTest/src/main/java/org/example/locationtest/MainActivity.java

```
/** 处理Google Play服务错误解决结果 */
@Override
protected void onActivityResult(int requestCode, int resultCode,
                                Intent data) {
   // 根据原始请求码决定处理方案
   switch (requestCode) {
      case CONNECTION_FAILURE_RESOLUTION_REQUEST:
         // 如果结果码为OK，尝试再次连接
         log("Resolution result code is: " + resultCode);
         switch (resultCode) {
            case Activity.RESULT_OK:
               // 尝试再次连接
               GoogleApiClient.connect();
               break;
         }
         break;
   }
}
```

每当活动启动另一个活动，并希望获得返回的结果时，都将调用这个方法。因此，首先必须检查请求码是否与`onConnectionFailed()`中设置的请求码相同。然后，必须检查解决方案是否管用（结果码为OK）。如果这两个条件都满足，就可以尝试再次连接。这次尝试应该可以成功。

12.2.6　请求权限

最后一步是，在文件AndroidManifest.xml中的标签`application`前面添加如下代码行。

locationTest/src/main/AndroidManifest.xml

```
<uses-permission
   android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission
```

```
    android:name="android.permission.ACCESS_FINE_LOCATION" />
```

这些代码告诉Android，应用需要访问用户的当前位置才能运行。

12.2.7 运行

所有方法都定义好后，编辑器窗口中应该没有任何错误。运行这个程序，你应该看到不断滚动的更新，如图12-2所示。请注意，这个程序需要有位置提供器才能运行，因此必须在实际设备上运行它。

注意到相邻日志项的时间间隔长短不一。位置服务会尽可能确保间隔为1000~5000毫秒（1~5秒），但并非总能成功。

如果你是在实际设备上运行该应用，请将设备拔下来，并带着它出去溜达一会儿。在你溜达时，将看到经度、纬度和精度在不断变化。

这些数字看起来可能很枯燥，但有了它们之后，就可以随便使用。为何不使用Maps API来绘制你的移动轨迹呢？也可以使用Geolocation API来显示你经过了哪些街道。甚至可以让应用在你接近喜欢的咖啡馆时提醒你。所有这些功能都可以使用Google API并编写少量代码来实现。

额外练习：让这个应用长时间运行。它将越来越慢，最终几乎没有反应。请找出原因并解决其中的问题。

12.3 快速阅读指南

本章简要地介绍了Google Play服务提供的激动人心的功能。我们创建了一个使用其中一种服务——位置服务的应用，但这只是皮毛。要更深入地了解Google Play服务，请参阅在线文档[①]。

本书还有一章就结束了，下一章将演示如何使用SQL在手机中存储结构化信息，如包含景点、照片和说明的旅行日志。

12

① http://d.android.com/google/play-services

第 13 章 使用SQL

过去30年来，数据库始终是企业应用程序开发的主要组成部分，但对小规模应用而言，它们太过昂贵、也太笨拙了。近年来，随着小型嵌入式引擎（如Android平台自带的引擎）的推出，情况发生了变化。

本章演示如何使用Android的嵌入式数据库引擎SQLite，你还将学习如何使用Android数据绑定功能将数据源关联到用户界面。你将学习`ContentProvider`类，它能够让两个应用共享数据。

可以使用内容提供器和SQLite在应用中存储地址列表、订单、棋谱以及众多其他的对象。需要在跨程序调用中传递大量数据项时，这些技术都将派上用场。

另外，本章还将演示如何使用`Loader`类来确保UI响应迅速而流畅。如果不使用加载器（loader），数据库访问和其他长时间运行的任务可能会导致用户界面停顿，给用户带来糟糕的体验。

13.1 SQLite 简介

SQLite[①]是一个功能强大的微型数据库引擎，由Richard Hipp博士于2000开发。SQLite无疑是全球部署最广泛的SQL数据库引擎。除Android外，其身影还出现在Apple iPhone、Symbian手机、Mozilla Firefox、Skype、PHP、Adobe AIR、Mac OS X、Solaris等众多地方。

SQLite许可

SQLite源代码不需要许可，因为它不受版权限制。SQLite源代码不但不需要许可，还会像下面这样祝福你。

愿你行善莫行恶。

愿你原谅自己、宽恕他人。

愿你宽心与人分享，所取不多于所施。

① http://www.sqlite.org

SQLite为何如此深受欢迎呢？下面是其中的3个原因。

- 它是免费的。作者放弃了版权，不对用户收取任何费用。
- 它很小。最新版本为150 KB，在Android手机的内存中容纳它绰绰有余。
- 它不需要安装和管理。不需要服务器，没有配置文件，不需要数据库管理员。

SQLite数据库就是一个文件。移动这个文件，甚至将其复制到其他系统（例如，从手机复制到工作站），它依然能够正常运行。Android将数据库文件存储在目录/data/data/packagename/databases中。可以使用命令`adb`来查看、移动和删除它。

要在程序中访问这个文件，需要运行结构化查询语言（SQL）语句，而不是调用Java I/O例程。Android通过辅助类和便利方法隐藏了一些SQL语法，但要使用SQLite，还是需要对SQL有个大致了解。

13.2 SQL 基础知识

如果你使用过Oracle、SQL Server、MySQL、DB2或其他数据库引擎，那么一定不会对SQL感到陌生，可以直接跳到13.3节；如若不然，则可通过本节对SQL大致有个了解。

为了使用SQL数据库，需要提交SQL语句并获取返回的结果。SQL语句分为三大类：DDL语句、修改语句和查询语句。

13.2.1 DDL语句

数据库文件可包含很多表。表由行组成，其中每行包含的列数都是固定的。表的每列都有名称和数据类型（文本字符串、数字等）。首先，需要运行数据定义语言（DDL）语句来定义这些表和列。例如，下面的语句用于创建一个包含3列的表。

sqlite/create.sql

```
create table mytable (
   _id integer primary key autoincrement,
   name text,
   phone text );
```

其中一列被指定为主键——唯一标识行的数字。`AUTOINCREMENT`意味着每添加一条记录，数据库就将主键加1，以确保每条记录都是唯一的。根据约定，第一列总是被命名为`_id`。SQLite并不要求表必须包含`_id`列，但后面使用Android内容提供器时将需要它。

请注意，不同于其他大多数数据库，在SQLite中，列类型只是提示。如果你试图在整型列中存储字符串或在字符串列中存储整型数据，不会导致任何错误。SQLite的作者将此视为一种特色，而非bug。

13.2.2　修改语句

SQL提供了很多让你能够在数据库中插入、删除和更新记录的语句。例如，要添加几个电话号码，可以使用如下代码。

sqlite/insert.sql

```
insert into mytable values(null, 'Steven King', '555-1212');
insert into mytable values(null, 'John Smith', '555-2345');
insert into mytable values(null, 'Fred Smitheizen', '555-4321');
```

值的排列顺序与`CREATE TABLE`语句中列的排列顺序相同。这里将`_id`列的值指定成了`NULL`，因为SQLite会为我们计算该列的值。

13.2.3　查询语句

在表中添加数据后，就可以使用`SELECT`语句对表执行查询了。例如，要获取第3条记录，可以采取如下做法。

sqlite/selectid.sql

```
select * from mytable where(_id=3);
```

你可能希望根据人名查询电话号码。下面演示如何查找所有姓名中包含Smith的记录。

sqlite/selectwhere.sql

```
select name, phone from mytable where(name like "%smith%");
```

别忘了SQL不区分大小写。无论是关键字、列名还是搜索字符串，都可指定为大写，也可指定为小写。

对SQL有足够的了解后，下面来看看如何在一个简单程序中应用这些知识。

13.3　一个简单的数据库程序

为演示SQLite，下面来创建一个简单应用——Events。它用于在数据库中存储记录，并显示这些记录。首先创建一个简单应用，然后不断添加功能。在新建项目向导中，使用如下设置来新建一个程序。

- 应用名：Events
- 公司域名：example.org
- 尺寸：Phone and Tablet
- 最低SDK：API 16: Android 4.1（Jelly Bean）
- 添加活动：Blank Activity
- 活动名：MainActivity

- 布局名：activity_main
- 标题：Events

与往常一样，你可从本书的配套网站[1]下载完整的源代码（查找示例eventsv1）。

这里需要存储一些描述数据库的常量，为此要创建一个`Constants`接口。

eventsv1/src/main/java/org/example/events/Constants.java

```java
package org.example.events;
import android.provider.BaseColumns;
public interface Constants extends BaseColumns {
   public static final String TABLE_NAME = "events";
   // 数据库中的列
   public static final String TIME = "time";
   public static final String TITLE = "title";
}
```

每个事件都将作为一行存储在`events`表中，每行都包含`_id`、`time`和`title`列。其中，`_id`是主键，它是在扩展的接口`BaseColumns`中定义的；`time`和`title`列将分别用于存储时间戳和事件名称。

13.3.1　使用SQLiteOpenHelper

接下来将创建一个名为`EventsData`的辅助类，用来表示数据库本身。它扩展了管理数据库创建和版本的Android类`SQLiteOpenHelper`。你需要做的只是提供一个构造函数，并重写两个方法。

eventsv1/src/main/java/org/example/events/EventsData.java

```java
package org.example.events;

import static android.provider.BaseColumns._ID;
import static org.example.events.Constants.TABLE_NAME;
import static org.example.events.Constants.TIME;
import static org.example.events.Constants.TITLE;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class EventsData extends SQLiteOpenHelper {
   private static final String DATABASE_NAME = "events.db";
   private static final int DATABASE_VERSION = 1;

   /** 创建一个表示Events数据库的辅助对象 */
   public EventsData(Context ctx) {
      super(ctx, DATABASE_NAME, null, DATABASE_VERSION);
   }
```

[1] http://pragprog.com/book/eband4

```
 -
20    @Override
 -    public void onCreate(SQLiteDatabase db) {
 -       db.execSQL("CREATE TABLE " + TABLE_NAME + " (" + _ID
 -             + " INTEGER PRIMARY KEY AUTOINCREMENT, " + TIME
 -             + " INTEGER," + TITLE + " TEXT NOT NULL);");
25    }
 -
 -    @Override
 -    public void onUpgrade(SQLiteDatabase db, int oldVersion,
 -          int newVersion) {
30       db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
 -       onCreate(db);
 -    }
 - }
```

构造函数始于第16行。其中，`DATABASE_NAME`是要使用的数据库的文件名（events.db），`DATABASE_VERSION`是虚构的编号。在实际程序中，每当对数据库设计做重大修改（如添加新列）时，都需要将版本号加1。

小乔爱问：

为何将`Constants`声明为接口？

这是 Java 的特色。我不知道你，我反正不喜欢每次使用常量时都必须加上类名。例如，我希望只输入 TIME，而不是 Constants.TIME。传统意义上，在 Java 中可以使用接口来实现这个目的。类可以继承接口 `Constants`，这样在引用该接口的任何字段时，都可以省略接口名。如果你查看接口 `BaseColumns`，就会发现开发 Android 的人员就使用了这种技巧。

然而，从 Java 5 起，出现了一种更好的办法：静态导入。在 `EventsData` 以及本章的其他类中，我就使用了这种办法。由于 `Constants` 是一个接口，因此可以使用新方法，也可以使用旧方法，这取决于你的喜好。

数据库首次被访问时，`SQLiteOpenHelper`会发现它是不存在的，需要调用方法`onCreate()`来创建它。第21行重写了方法`onCreate()`，使其运行一条`CREATE TABLE` SQL语句。这将创建表`events`以及包含它的数据库文件events.db。

当Android（根据版本号）确定你引用的是既有数据库时，将调用方法`onUpgrade()`（第28行）。在这个示例中，只实现了将旧表删除的操作，如果你愿意，也可以在这个方法中采取更聪明的做法。例如，可以运行一个`ALTER TABLE` SQL命令，在既有数据库中添加列。

13.3.2　定义主程序

编写程序Events时，一开始将使用本地SQLite数据库来存储事件，并将它们以字符串的方式

显示在TextView中。

采取如下方式定义布局文件layout/activity_main.xml。

eventsv1/src/main/res/layout/activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   android:paddingBottom="@dimen/activity_vertical_margin"
   tools:context=".MainActivity">
   <TextView
      android:id="@+id/text"
      android:layout_width="match_parent"
      android:layout_height="wrap_content" />
</ScrollView>
```

这里声明了一个TextView，给它指定了一个富有想像力的ID——text（在代码中为R.id.text），再将它放在一个ScrollView中，以防事件太多而无法在屏幕上同时显示出来。图13-1所示的屏幕截图说明了这个布局的效果。

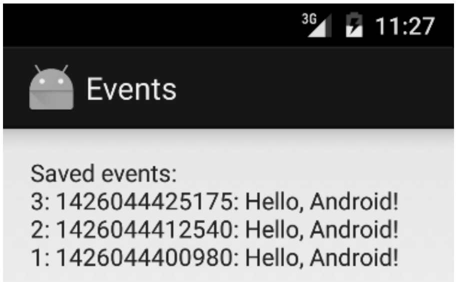


图　13-1

主程序为活动MainActivity的方法onCreate()，其轮廓如下。

eventsv1/src/main/java/org/example/events/MainActivity.java

```java
package org.example.events;

import static android.provider.BaseColumns._ID;
import static org.example.events.Constants.TABLE_NAME;
import static org.example.events.Constants.TIME;
import static org.example.events.Constants.TITLE;
import android.app.Activity;
import android.content.ContentValues;
import android.database.Cursor;
```

```
10 import android.database.sqlite.SQLiteDatabase;
 - import android.os.Bundle;
 - import android.widget.TextView;
 -
 - public class MainActivity extends Activity {
15    private EventsData events;
 -    @Override
 -    public void onCreate(Bundle savedInstanceState) {
 -       super.onCreate(savedInstanceState);
 -       setContentView(R.layout.activity_main);
20       events = new EventsData(this);
 -       try {
 -          addEvent("Hello, Android!");
 -          Cursor cursor = getEvents();
 -          showEvents(cursor);
25       } finally {
 -          events.close();
 -       }
 -    }
 - }
```

在方法onCreate()中，第19行用来设置视图的布局。第20行用来创建一个EventsData类的实例，然后是一个try块。如果你往下看第26行，将发现在finally块中关闭了数据库。这样，即便中途发生错误，数据库也会被关闭。

如果不包含任何事件，events表就没有多大意义了，因此第22行调用方法addEvent()在这个表中添加了一个事件。每次运行这个程序时，都将添加一个事件。如果你愿意，也可以在用户选择菜单、执行手势或按键时添加相应的事件。我把这项任务作为练习留给你去完成。

第23行调用方法getEvents()获取事件列表。最后，第24行调用方法showEvents()向用户显示事件列表。

非常简单，不是吗？下面来定义刚才调用的新方法。

13.3.3　添加记录

方法addEvent()在数据库中负责添加一条新记录，并将提供的字符串用作事件名。

eventsv1/src/main/java/org/example/events/MainActivity.java

```
private void addEvent(String string) {
   // 在数据源Events中插入一条新记录
   // 你将执行类似的操作来删除和更新记录
   SQLiteDatabase db = events.getWritableDatabase();
   ContentValues values = new ContentValues();
   values.put(TIME, System.currentTimeMillis());
   values.put(TITLE, string);
   db.insertOrThrow(TABLE_NAME, null, values);
}
```

由于需要修改数据，因此调用方法getWritableDatabase()获取一个指向数据库events的读写句柄。该数据库句柄会被缓存起来，因此想调用这个方法多少次都可以。

接下来，使用当前时间和事件名填充一个ContentValues对象，并将该对象传递给方法insertOrThrow()，让它执行实际的INSERT SQL语句。这里不需要传入记录ID，因为SQLite会自动创建并返回一个ID。

顾名思义，如果失败，insertOrThrow()将引发类型为SQLException的异常。无需使用关键字throws来声明这种异常，因为它是RuntimeException，而不是checked异常。然而，如果你愿意，也可以像处理其他异常那样在try/catch块中处理它。发生错误时，如果不进行任何处理，程序将终止，并将栈跟踪写入Android日志。

默认情况下，执行插入操作后，数据库将立即更新。如果要以批量方式进行修改或延迟修改，请参阅SQLite网站了解详情。

13.3.4　运行查询

方法getEvents()负责执行数据库查询以获取事件列表。

eventsv1/src/main/java/org/example/events/MainActivity.java

```
private static String[] FROM = { _ID, TIME, TITLE, };
private static String ORDER_BY = TIME + " DESC";
private Cursor getEvents() {
   // 执行托管查询（managed query），活动会处理关闭工作，
   // 并在必要时重新查询游标
   SQLiteDatabase db = events.getReadableDatabase();
   Cursor cursor = db.query(TABLE_NAME, FROM, null, null, null,
         null, ORDER_BY);
   startManagingCursor(cursor);
   return cursor;
}
```

查询时不需要修改数据库，因此调用方法getReadableDatabase()来获取一个只读句柄。接下来，调用query()来执行实际的SELECT SQL语句。FROM是一个数组，指定了要检索的列，ORDER_BY能够让SQLite按从新到旧的顺序返回结果。

方法query()还有用于指定WHERE子句、GROUP BY子句和HAVING子句的参数，但这个示例没有使用它们。实际上，query()的目标就是为程序员提供方便。如果你愿意，也可以以字符串的方式创建SELECT语句，然后再调用方法rawQuery()来执行它。这两个方法都返回一个表示结果集的Cursor对象。

Cursor类似于Java Iterator和JDBC ResultSet。可以对其调用相应的方法来获取有关当前记录的信息，然后调用另一个方法来移到下一条记录。稍后在显示结果时，你将看到使用Cursor的方法。

最后一步是调用startManagingCursor()，它能让活动根据自己的生命周期来管理游标的生命周期。例如，活动暂停时，程序会自动将游标除活；而活动重新启动时，程序又将重新查询游标；活动终止时，程序将关闭所有托管的游标。

请注意，在较新的Android版本中，startManagingCursor()已经被摒弃了。不过它依然管用，只是还有更佳的方式，这将在13.7节介绍。

13.3.5　显示查询结果

需要定义的最后一个方法是showEvents()，它会将一个Cursor对象作为参数，并设置输出的格式，以方便用户查看。

eventsv1/src/main/java/org/example/events/MainActivity.java

```
Line 1 private void showEvents(Cursor cursor) {
     -    // 将要显示的内容都塞到一个字符串中
     -    StringBuilder builder = new StringBuilder(
     -          "Saved events:\n");
     5    while (cursor.moveToNext()) {
     -       // 也可以使用getColumnIndexOrThrow()来获取列索引
     -       long id = cursor.getLong(0);
     -       long time = cursor.getLong(1);
     -       String title = cursor.getString(2);
    10       builder.append(id).append(": ");
     -       builder.append(time).append(": ");
     -       builder.append(title).append("\n");
     -    }
     -    // 显示到屏幕上
    15    TextView text = (TextView) findViewById(R.id.text);
     -    text.setText(builder);
     - }
```

在Events示例的这个版本中，创建了一个大型字符串（第3行）来存储所有的事件记录，并用换行符分隔记录。不推荐采用这种做法，不过目前它还是可行的。

第5行调用方法Cursor.moveToNext()移到数据集中的下一条记录。刚获取的Cursor对象指向第一条记录的前面，因此调用moveToNext()将移到第一条记录。不断循环，直到moveToNext()返回false（表明后面没有其他记录）为止。

在循环中，调用getLong()和getString()获取目标列中的数据（第7~9行），再将它们附加到字符串的末尾（第10~12行）。Cursor类还有一个getColumnIndexOrThrow()方法。原本可以使用它来获取列索引号（传递给getLong()和getString()的值0、1和2）。然而，这个方法的速度有点慢，因此如果需要，应在循环外调用它，并将返回的索引号存储到变量中。

处理完所有记录后，获取layout/activity_main.xml中定义的TextView，并将前面创建的大型字符串加入其中，如第15~16行所示。

如果现在运行这个示例，结果将类似于图13-1。祝贺你，已经创建了第一个Android数据库程序！但它需要改进的地方有很多。

如果有数千乃至数百万个事件，结果将如何呢？程序的运行速度将非常慢，且创建包含所有这些事件的字符串时，可能会耗尽内存。如果要让用户能够选择并处理一个事件，该怎么办呢？如果将全部事件都放在一个字符串中，就无法提供这样的功能。好在Android 提供了更佳的方式：数据绑定。

13.4 数据绑定

数据绑定只需让你编写几行代码就能将模型（数据）关联到视图。为了演示数据绑定，将对前面的Events示例进行修改，在其中使用一个`ListView`，并将其绑定到数据库查询结果。为此，首先让`MainActivity`类扩展ListActivity（而不是Activity）。

eventsv2/src/main/java/org/example/events/MainActivity.java

```
import android.app.ListActivity;
// ……
public class MainActivity extends ListActivity {
   // ……
}
```

接下来，在方法`MainActivity.showEvents()`中修改事件的显示方式。

eventsv2/src/main/java/org/example/events/MainActivity.java

```
import android.widget.SimpleCursorAdapter;
// ……
   private static int[] TO = { R.id.rowid, R.id.time, R.id.title, };
   private void showEvents(Cursor cursor) {
      // 设置数据绑定
      SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
            R.layout.item, cursor, FROM, TO);
      setListAdapter(adapter);
   }
```

注意到，这里的代码比以前少得多（两行而不是10行）。第1行根据`Cursor`创建一个`SimpleCursorAdapter`，第2行让ListActivity使用这个新的适配器。适配器充当了中间人的角色，将视图和数据源关联了起来。

你可能还记得，首次使用适配器是在第11章的Suggest示例程序中。在那个示例中，使用了一个`ArrayAdapter`，因为数据源是在XML中定义的数组。在这个示例中，使用的是`SimpleCursorAdapter`，因为数据源是数据库查询返回的`Cursor`对象。

`SimpleCursorAdapter`的构造函数接受以下5个参数。

- `context`：指向当前活动的引用。

13

- layout：为单个列表项定义视图的资源。
- cursor：数据集游标。
- from：提供数据的列列表。
- to：显示数据的视图列表。

这个方法在较新的Android版本中已经被摒弃了，但目前依然管用。本章后面将探索其替代品。

列表项的布局是在layout/item.xml中定义的。请注意数组TO引用的视图rowid、time和title的定义。

eventsv2/src/main/res/layout/item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="horizontal"
   android:padding="10sp">
   <TextView
      android:id="@+id/rowid"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
   <TextView
      android:id="@+id/rowidcolon"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text=": "
      android:layout_toRightOf="@id/rowid" />
   <TextView
      android:id="@+id/time"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_toRightOf="@id/rowidcolon" />
   <TextView
      android:id="@+id/timecolon"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text=": "
      android:layout_toRightOf="@id/time" />
   <TextView
      android:id="@+id/title"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:ellipsize="end"
      android:singleLine="true"
      android:textStyle="italic"
      android:layout_toRightOf="@id/timecolon" />
</RelativeLayout>
```

该布局看起来比实际情况复杂。它只是将ID、时间和名称放在一行，并用冒号分隔它们。为了让结果看起来更漂亮，我指定了较小的内边距，并设置了格式。

最后，需要在layout/activity_main.xml中修改活动本身的布局。修改后的版本如下。

eventsv2/src/main/res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">
   <!-- 请注意表示“列表”和“空”的内置id -->
   <ListView
      android:id="@android:id/list"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"/>
   <TextView
      android:id="@android:id/empty"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/empty" />
</LinearLayout>
```

由于这个活动扩展了ListActivity，Android将在布局文件中查找两个特殊的id。如果列表不为空，将显示视图android:id/list；否则显示视图android:id/empty。这样，没有列表项时，用户看到的将是消息“No events!”，而不是空屏幕。

下面是需要的字符串资源。

eventsv2/src/main/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name">Events</string>
   <string name="action_settings">Settings</string>
   <string name="empty">No events!</string>
</resources>
```

最终的结果如图13-2所示。

作为练习，请你想想如何改进这个使用了列表的应用。例如，用户选择一个事件时，可以打开一个详细视图、通过邮件将事件发送给技术支持或将选定事件及其后面的所有事件都从数据库中删除。

这个示例还有一个小问题：其他应用无法写入事件数据库，甚至都无法查看其内容！为了解决这个问题，需要使用Android内容提供器。

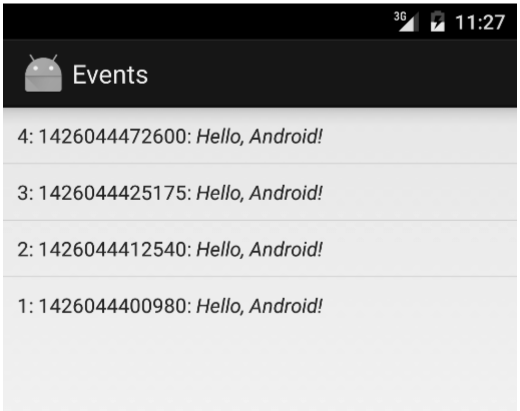


图13-2　最终结果——使用ListActivity和数据绑定

13.5　使用内容提供器

在Android安全模型中（请参阅2.4节的讨论），一个应用写入的文件不能被其他应用读取或写入。每个程序都有自己的Linux用户ID、数据目录（/data/data/packagename）和受保护的内存空间。Android程序可以如下两种方式相互通信。

- 进程间通信（IPC）：进程使用Android接口定义语言（AIDL）和`IBinder`接口定义API。调用API时，会在进程之间安全而高效地封送（marshal）参数。这种高级技术用于远程调用后台服务线程。IPC、服务和binder不在本书的讨论范围内。要更详细地了解它们，请参阅有关aidl[①]、服务[②]和IBinder[③]的在线文档。
- 内容提供器：在系统中将进程注册为特定数据的提供器。需要这些信息时，Android通过固定的API调用进程，以进程认为合适的方式查询或修改内容。将在Events示例中使用这种技术。

内容提供器管理的信息都使用类似于下面的URI进行编址。

```
content://authority/path/id
```

其中各个组成部分的含义如下。

- `content://`是相应标准指定的必不可少的前缀。
- `authority`为提供器的名称。为避免名称冲突，建议使用全限定包名。
- `path`是提供器中的虚拟目录，标识了请求的数据类型。

① http://d.android.com/guide/components/aidl.html
② http://d.android.com/reference/android/app/Service.html
③ http://d.android.com/reference/android/os/IBinder.html

- id是请求的特定记录的主键。要请求特定类型的所有记录，可省略该id和末尾的斜杠。

Android提供了多种内置提供器，其中包括：

- content://browser
- content://contacts
- content://media
- content://settings

要获悉最新的内置提供器清单，请参阅在线文档[①]。访问这些提供器时，不使用上述字符串，而使用Browser.BOOKMARKS_URI等常量。请注意，在访问某些提供器时，必须在清单文件中请求额外的权限。

为了演示提供器的用法，下面修改Events示例，在其中使用提供器。对于我们的事件提供器，下面是一些有效的URI。

```
content://org.example.events/events/3  -- _id为3的单个事件
content://org.example.events/events  -- 所有事件
```

首先，需要在Constants.java中添加两个常量。

eventsv3/src/main/java/org/example/events/Constants.java

```java
import android.net.Uri;
// ……
   public static final String AUTHORITY = "org.example.events";
   public static final Uri CONTENT_URI = Uri.parse("content://"
         + AUTHORITY + "/" + TABLE_NAME);
```

不需要对布局文件（activity_main.xml和item.xml）进行修改，因此下一步是对MainActivity类做一些细微的修改。

13.5.1 修改主程序

主程序（方法MainActivity.onCreate()）实际上更简单，因为不需要跟踪数据库对象。

eventsv3/src/main/java/org/example/events/MainActivity.java

```java
@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   addEvent("Hello, Android!");
   Cursor cursor = getEvents();
   showEvents(cursor);
}
```

① http://d.android.com/reference/android/provider/package-summary.html

这里不再需要try/finally块，并删除了指向EventData的引用。

13.5.2　添加记录

在addEvent()中，有两行代码是不同的，新版本如下。

eventsv3/src/main/java/org/example/events/MainActivity.java

```
import static org.example.events.Constants.CONTENT_URI;
// ……
   private void addEvent(String string) {
      // 在数据源Events中插入一条新记录
      // 你将执行类似的操作来删除和更新记录
      ContentValues values = new ContentValues();
      values.put(TIME, System.currentTimeMillis());
      values.put(TITLE, string);
      getContentResolver().insert(CONTENT_URI, values);
   }
```

这里不再需要调用getWritableDatabase()，此外，应调用getContentResolver().insert()而不是insertOrThrow()，并传入一个内容URI而不是数据库句柄。

13.5.3　运行查询

使用内容提供器时，方法getEvents()也更简单了。

eventsv3/src/main/java/org/example/events/MainActivity.java

```
private Cursor getEvents() {
   // 执行托管查询。活动会处理关闭工作，
   // 并在必要时重新查询游标
   return managedQuery(CONTENT_URI, FROM, null, null, ORDER_BY);
}
```

这里调用了方法Activity.managedQuery()，并传入了内容URI、感兴趣的列列表以及排序方式。

通过删除所有数据库引用，实现了Events客户端和Events数据提供器之间的解耦。客户端变得更简单了。但是，接下来必须实现以前没实现过的东西。

13.6　实现内容提供器

内容提供器是一种高级对象，与活动一样，必须向系统声明。因此，要实现内容提供器，首先必须在文件AndroidManifest.xml中声明它，即在标签<application>中标签<activity>的前面添加如下标签。

eventsv3/src/main/AndroidManifest.xml

```
<provider
      android:name=".EventsProvider"
      android:authorities="org.example.events"/>
```

其中，android:name是类名（附加在包名后面），android:authorities是用于内容URI中的字符串。

接下来，创建EventsProvider类。这个类必须扩展ContentProvider，其基本轮廓如下。

eventsv3/src/main/java/org/example/events/EventsProvider.java

```
package org.example.events;

import static android.provider.BaseColumns._ID;
import static org.example.events.Constants.AUTHORITY;
import static org.example.events.Constants.CONTENT_URI;
import static org.example.events.Constants.TABLE_NAME;
import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.text.TextUtils;

public class EventsProvider extends ContentProvider {
   private static final int EVENTS = 1;
   private static final int EVENTS_ID = 2;

   /** 事件目录的MIME类型 */
   private static final String CONTENT_TYPE
      = "vnd.android.cursor.dir/vnd.example.event";

   /** 单个事件的MIME类型 */
   private static final String CONTENT_ITEM_TYPE
      = "vnd.android.cursor.item/vnd.example.event";

   private EventsData events;
   private UriMatcher uriMatcher;
   // ……
}
```

根据约定，要在MIME类型中使用vnd.example而不是org.example。多用途Internet邮件扩展类型（MIME，Multipurpose Internet Mail Extensions）是一个描述内容类型的Internet标准。

EventsProvider可以处理两种类型的数据。

- EVENTS（MIME类型CONTENT_TYPE）：事件目录或事件列表。
- EVENTS_ID（MIME类型CONTENT_ITEM_TYPE）：单个事件。

就URI而言，差别在于第一种类型不指定ID，而第二种类型可以指定。使用Android类`UriMatcher`来分析URI，以确定客户端指定的是哪种类型。在这个提供器中，我们重用了本章前面的`EventsData`类来管理实际数据库。

为了节省篇幅，这里没有列出这个类的其他代码，你可以从本书的配套网站下载其完整源代码。Events示例的所有版本都可在源代码ZIP文件中找到。

从外部看，Events示例的这个版本与前一个版本完全相同，但在内部，它提供了一个事件存储框架，可供系统中的其他应用，乃至其他开发人员编写的应用使用。

13.7　使用加载器

编写本章示例的代码时，你可能已经注意到，有警告指出我们使用的方法`managedQuery()`和构造函数`SimpleCursorAdapter`都已经被摒弃了。不过它们依然管用，网上有很多示例都在使用它们。然而，在执行长时间运行的操作方面，最新的Android版本提供了一种好得多的方式：使用加载器。

使用加载器的方式更佳的原因有多个。

- 它们以异步方式加载数据。`managedQuery()`在UI线程中加载数据，可能会导致用户界面停顿或断断续续。
- 它们监视数据源，在数据可用时立即提供结果，而不要求重新查询。
- 它们在设备配置发生变化时保留数据，避免了用户旋转屏幕或暂停应用时重新进行查询。

为了使用加载器，必须稍微调整本章应用的结构。所幸需要做的所有修改都是在`MainActivity`类中进行的。首先，在开头添加几条`import`语句，以导入所需的类。

eventsv4/src/main/java/org/example/events/MainActivity.java

```
import android.app.LoaderManager;
import android.content.CursorLoader;
import android.content.Loader;
```

接下来，修改`MainActivity`类，使其实现接口`LoaderCallbacks`，并声明两个新的变量。

eventsv4/src/main/java/org/example/events/MainActivity.java

```
public class MainActivity extends ListActivity implements
      LoaderManager.LoaderCallbacks<Cursor> {
   // ……
   // 当前活动中加载器独一无二的ID
   private final static int LOADER_ID = 1;

   // 将数据绑定到ListView的适配器
   private SimpleCursorAdapter mAdapter;
}
```

然后，删除方法getEvents()和showEvents()，因为已经不再需要它们了。将方法onCreate()修改成如下形式。

eventsv4/src/main/java/org/example/events/MainActivity.java

```
@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);

   // 初始化适配器。它一开始为空
   mAdapter = new SimpleCursorAdapter(this, R.layout.item, null, FROM, TO, 0);

   // 将适配器关联到ListView
   setListAdapter(mAdapter);

   // 初始化加载器
   LoaderManager lm = getLoaderManager();
   lm.initLoader(LOADER_ID, null, this);

   addEvent("Hello, Android!");
}
```

在方法onCreate()中创建适配器，并将其关联到列表视图，而不是等到以后再这样做。这个适配器一开始为空，但后面将使用方法addEvent()在其中添加一个事件。接下来，获取一个指向加载器管理器的句柄，并初始化一个加载器。

给方法initLoader()传递的最后一个参数为this，它指向刚才修改为实现了接口LoaderCallbacks的MainActivity类。因此，接下来需要实现这个接口定义的3个必不可少的方法。

eventsv4/src/main/java/org/example/events/MainActivity.java

```
@Override
public Loader<Cursor> onCreateLoader(int id, Bundle args) {
   // 创建一个CursorLoader对象
   return new CursorLoader(this, CONTENT_URI, FROM, null, null, ORDER_BY);
}

@Override
public void onLoadFinished(Loader<Cursor> loader, Cursor cursor) {
   switch (loader.getId()) {
      case LOADER_ID:
         // 数据可用
         mAdapter.swapCursor(cursor);
         break;
   }
}

@Override
public void onLoaderReset(Loader<Cursor> loader) {
   // 加载器的数据不可用
```

```
    mAdapter.swapCursor(null);
}
```

第一个方法（onCreateLoader()）在需要创建新的加载器时被调用，它只负责创建并返回一个新的CursorLoader对象。

方法onLoadFinished()在加载器填充好游标时被调用。游标填充好后，就可以将其换入适配器，让适配器（及其关联的列表）开始显示新数据。

最后，方法onLoaderReset()在加载器中没有数据时被调用，因为必须用不包含任何数据的空游标换出原来的游标。

现在运行这个程序时，外观几乎与以前完全相同。这个应用的运行速度原本就非常快，因此几乎看不出有什么差别。但请相信我，要创建流畅而不是断断续续的应用，务必将数据库查询、网络I/O、计算以及任何耗时的操作移出主GUI线程。

13.8　快速阅读指南

在本章中，你学习了如何在Android SQL数据库中存储数据。要使用SQL实现更多的功能，必须学习本章未介绍的其他语句和表达式。为此，购买Jonathan Gennick编著的*SQL Pocket Guide* [Gen10]或Mike Owens编写的《SQLite权威指南》（*The Definitive Guide to SQLite*）[AO10]会是不错的选择。但别忘了，在不同的数据库引擎中，SQL语法和函数存在细微的差别。

可以对本章介绍的SimpleCursorAdapter进行定制，以显示除文本外的其他内容。例如，可根据Cursor对象中的数据显示星级、迷你图（小型图表）或其他视图。有关这方面的更详细信息，请参阅SimpleCursorAdapter文档[①]中的ViewBinder。

使用加载器并非改善应用性能的唯一途径。例如，在第11章中，我们就使用了Executor和Runnable类在独立的线程中执行网络I/O操作。

对Android的简要介绍到这里就结束了。为了让你接下来的旅程更加顺利，务必查看本书配套网站[②]提供的资源和源代码。另外，在Stack Overflow网站[③]的Android频道，还可找到很多问题的答案。

现在就去打造卓越的应用吧！

① http://d.android.com/reference/android/widget/SimpleCursorAdapter.html
② http://pragprog.com/book/eband4
③ http://stackoverflow.com/questions/tagged/android

第五部分

附　　录

附录

Java和Android在语言和API方面的异同

从很大程度上说，Android程序是使用Java语言编写的，使用的是Java 6 Standard Edition（SE）库API。这里之所以说“从很大程度上说”，是因为还存在一些不同之处。本附录介绍常规Java和Android使用的Java之间的差别。ART（Android 5.0和更高版本使用的运行时系统）和Dalvik（Android 5.0之前的版本使用的运行时系统）也存在这样的限制。

如果你熟悉其他平台上的Java开发，应仔细研究本附录，以了解需要将哪些功能抛诸脑后。

A.1 语言子集

Android使用标准Java编译器将源代码编译成常规的字节码，然后将这些字节码转换为要执行的指令。因此，Android几乎全面支持Java语言，而不仅仅是其一小部分。通过使用编译器和字节码，在应用中使用库时，通常不需要获取其源代码。

A.1.1 语言等级

Android Studio支持与Java Standard Edition 6或更早版本兼容的代码，还可支持Java 7新增的功能，但当前不支持任何Java 8功能。

要支持Java 7语言功能，需要在gradle编译文件中添加如下代码行。

```
compileOptions {
   sourceCompatibility JavaVersion.VERSION_1_7
   targetCompatibility JavaVersion.VERSION_1_7
}
```

这能够让你在针对Android 2.2（Froyo）和更高版本的应用中使用Java 7的如下语言功能：

- 钻石运算符（<>）
- 在switch语句中使用字符串
- 在一条catch语句中捕获多种异常（catch (Exc1 | Exc2 e)）

- 在数字字面量中使用下划线（`1_234_567`）
- 二进制字面量（0b1110111）

这还能够让你在针对Android 4.4（KitKat）和更高版本的应用中使用如下功能：

- `try-with-resources`语句
- `@SafeVarargs`注解

当然，要解锁这些功能，必须安装Java 7或更高版本的JDK。

A.1.2　内置类型

支持所有的Java内置类型，包括`byte`、`char`、`short`、`int`、`long`、`float`、`double`、`Object`、`String`和数组。在较旧的低端硬件上，可能会模拟浮点数数学运算。这意味着，涉及`double`或`float`的运算将在软件而不是硬件中执行，因此速度比整数算术运算慢得多。然而，较新的Android设备都装备了优秀的浮点硬件，因此这通常不是问题。

A.1.3　多线程和同步

多线程是通过时隙技术实现的。依次给每个线程若干毫秒的运行时间，然后执行上下文切换，让另一个线程执行。虽然Android支持任意数量的线程，但一条经验规则是，线程数不应超过设备的处理器核数加1。一个线程专用于主用户界面（如果有的话），其他线程用于执行长时间运行的操作，如计算或网络I/O。

例如，在四核设备上运行时，将总线程数限制为5，可让应用获得最佳的性能。要获悉设备的处理器核数，可使用函数`Runtime.getRuntime().availableProcessors()`。

Android实现了关键字`synchronized`以及与同步相关的库方法，如`Object.wait()`、`Object.notify()`和`Object.notifyAll()`。它还为支持复杂算法提供了java.util.concurrent包。要让多个线程互不干扰，可像在其他Java程序中那样使用它们。

A.1.4　反射

虽然Android平台支持Java反射，但并不应该使用它，尤其是在对时间敏感的代码中。其中的原因很简单：为了改善性能和节省电量。因为反射的速度很慢，而速度缓慢的代码会消耗更多的电量。可以考虑使用替代品，如编译阶段工具和预处理器。

A.1.5　终结

像常规Java VM一样，Android支持在垃圾收集阶段终结对象。然而，大多数Java专家都建议不要依赖于终结器（finalizer），因为它们是否会运行以及何时运行都是不确定的。应使用显式方

法`close()`或`terminate()`。Android用于资源有限的硬件中，因此必须立即释放不再需要的资源。这一点很重要。

A.2 标准库子集

Android支持一个相对较大的Java Standard Edition 6.0库子集。有些API不支持是因为它们不适用，还有一些不支持是因为有更好的专用于Android的API。前面说过，Android 4.4（KitKat）和更高版本支持一些Java 7新增的功能，包括`AutoClosable`接口和`@SafeVarargs`注解。然而，它们通常不支持Java 7和Java 8新增的库API，包括新的文件系统API（NIO 2.0）、Fork和Join以及invokedynamic。

A.2.1 支持的包

Android支持如下标准包。要获悉如何使用它们，请参阅Java 2 Platform Standard Edition 6.0 API文档[1]。

- java.awt.font：一些与Unicode和字体相关的常量。
- java.beans：一些修改JavaBeans属性的类和接口。
- java.io：文件和流I/O。
- java.lang（java.lang.management除外）：语言和异常支持。
- java.math：大数、圆整和精度。
- java.net：网络I/O、URL和套接字。
- java.nio：文件和通道I/O。
- java.security：授权、证书和公钥。
- java.sql：数据库接口。
- java.text：格式设置、自然语言和排序规则（collation）。
- java.util（包括java.util.concurrent）：列表、映射、集、数组和集合。
- javax.crypto：加密算法和公钥。
- javax.microedition.khronos：OpenGL图形（来自Java Micro Edition）。
- javax.net：套接字工厂和SSL。
- javax.security（javax.security.auth.kerberos、javax.security.auth.spi和javax.security.sasl除外）。
- javax.sql（javax.sql.rowset除外）：数据库接口。
- javax.xml.parsers：XML分析。
- org.w3c.dom（不包括子包）：DOM节点和元素。
- org.xml.sax：Simple API for XML。

① http://docs.oracle.com/javase/6/docs/api/

请注意，虽然也支持常规Java SQL数据库API（JDBC），但不应使用它们来访问本地SQLite数据库，而使用android.database API（参见第13章）。

A.2.2　不支持的包

Android不支持Java 2 Platform Standard Edition中的下述包：

- java.applet
- java.awt
- java.lang.management
- java.rmi
- javax.accessibility
- javax.activity
- javax.imageio
- javax.management
- javax.naming
- javax.print
- javax.rmi
- javax.security.auth.kerberos
- javax.security.auth.spi
- javax.security.sasl
- javax.sound
- javax.swing
- javax.transaction
- javax.xml（javax.xml.parsers除外）
- org.ietf.*
- org.omg.*
- org.w3c.dom.*（子包）

A.3　第三方库

为方便开发人员使用，除前面列出的标准库外，Android SDK还自带了多个第三方库。

- org.json：JavaScript对象表示法（JavaScript Object Notation）。
- org.xml.sax：XML分析。
- org.xmlpull.v1：XML分析。

参考文献

[AGH05] Ken Arnold，James Gosling，David Holmes. Java程序设计语言[M]. 陈昊鹏，章程，张思博，等，译. 北京：人民邮电出版社，2006.

[AO10] Grant Allen，Mike Owens. SQLite权威指南. 杨谦，刘义宣，谢志强，译. 第2版. 北京：电子工业出版社，2012.

[Blo08] Joshua Bloch. *Effective Java*（中文版）. 杨春花，俞黎敏，译. 第2版. 北京：机械工业出版社，2009.

[EF14] Ben Evans，David Flanagan. Java技术手册（第6版）[M]. 6th ed. Sebastopol：O'Reilly & Associates，Inc.，2014.

[Gen10] Jonathan Gennick. *SQL Pocket Guide*[M]. 3rd ed. Sebastopol：O'Reilly & Associates, Inc.，2010.

[Goe06] Brian Goetz. JAVA并发编程实战[M]. 童云兰，等，译. 北京：机械工业出版社，2012.

[HT99] Andrew Hunt，David Thomas. 程序员修炼之道：从小工到专家[M]. 马维达，译. 北京：电子工业出版社，2010.

[JT95] Ollie Johnston，Frank Thomas. *The Illusion of Life: Disney Animation*[M]. New York：Disney Editions，1995.

[SB05] Kathy Sierra，Bert Bates. *Head First Java*（中文版）[M]. O'Reilly Taiwan公司，译. 张然，等，改编. 第2版. 北京：中国电力出版社，2007.

[Ses05] Peter Sestoft. *Java Precisely*[M]. 2nd ed. London：MIT Press, Cambridge，2005.

“Ed再次向初中级Android开发人员推出了一部精巧的学习指南。这部指南实用且引人入胜，适合刚步入Android应用开发领域的人员阅读，也可供有一定经验，欲更深入地了解基本游戏开发、动画、音效、线程、数据库和Google Play服务等概念的人员参考。”

——Diego Torres Milano，技术发烧友、Android系统工程师、Linux拥趸、作者

“在引领读者初识Android应用开发方面，其他图书难以与之比肩！”

——Mark Murphy，CommonsWare创始人，*The Busy Coder's Guide to Android Development* 的作者

“以令人愉悦的写作风格，通过引人入胜的示例，简明扼要地阐述了大量基础知识，适合所有想快速掌握Android开发的人员阅读。”

——Jason Pike，theswiftlearner.com软件开发人员

“介绍Play Store的一章表明，将应用提交到这个应用商店易如反掌，真是令人醍醐灌顶。”

——Stephen Wolff，Max Gate Digital有限公司董事

“这本书非常易读易懂，有关于正确编程方法的精美图片、细节和小提示。时至今日，我想它仍然是关于Android的优秀图书之一。这本书不是给编程新手看的，而是给Android编程新手看的。”

——Amazon.com读者评论

“最开始拿到这本书时我其实并不知道其中的内容，是非常技术性还是连门外汉也能看懂？令我吃惊的是它二者兼具。出于一些原因，我十分想开发出自己的Android应用程序。这本书帮我奠定了基础，而且借助示例一步步地讲解，让整个开发过程易于理解和实践。”

——Amazon.com读者评论

The Pragmatic Programmers

图灵社区：iTuring.cn
热线：(010)51095186转600

分类建议 计算机/程序设计/Android

人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-40860-0

ISBN 978-7-115-40860-0
定价：49.00元

看完了

如果您对本书内容有疑问，可发邮件至contact@turingbook.com，会有编辑或作译者协助答疑。也可访问图灵社区，参与本书讨论。

如果是有关电子书的建议或问题，请联系专用客服邮箱：ebook@turingbook.com。

在这里可以找到我们：

微博 @图灵教育 : 好书、活动每日播报
微博 @图灵社区 : 电子书和好文章的消息
微博 @图灵新知 : 图灵教育的科普小组
微信 图灵访谈 : ituring_interview，讲述码农精彩人生
微信 图灵教育 : turingbooks